A GPSS-FORTRAN CASE STUDY

by

Charles Joseph Petronis

United States
Naval Postgraduate School

THESIS

A GPSS-FORTRAN CASE STUDY

by

Charles Joseph Petronis

April 1970

A GPSS-FORTRAN Case Study

INTERNALLY DISTRIBUTED by REPORT

Charles Joseph Petronis
Major, United States Army
B.S., Norwich University, 1960

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
April, 1970

ABSTRACT

The objective of this study was to compare two commonly used computer simulation languages: GPSS and FORTRAN. The comparison was made by simulating an identical system in both languages. The comparison criteria used to evaluate the languages were as follows: ability to represent system, simulation time, ability to represent stocastic phenomena, programming time, computer running time, monitoring and debugging, storage requirements, data initialization and starting conditions, replication, data collection and display. The results from this study indicated that a system may be modeled four to five times faster using GPSS as compared to FORTRAN. The modeled system was simpler to conceptualize in GPSS, and the model required reduced programming, debugging and monitoring time compared to the FORTRAN model.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ACKNOWLEDGEMENT

The author wishes to express sincere appreciation to Professor Alvin F. Andrus for supervision, encouragement and for the many suggestions that materially added to this study.

The author especially wishes to thank his wife, Bunny, and daughters, Debra, Susan, and Karen for administrative assistance and for the many sacrifices they made during the writing of this thesis.

I. INTRODUCTION

Computer simulation has become an ordinary and useful technique of the operations analyst. In industrial and military operations research, complex as well as simple systems are being simulated by computer for both educational and analytical purposes. In an effort to reduce both the amount of detailed programming required and the amount of intricate simulation methodology that must be known in order to create a computer simulation, several special purpose simulation languages have been made available to the operations research community [Refs. 15, 16, 17]. Prior to the introduction of these simulation languages, computer simulation programs were restricted to either the available assembly or compiler languages. It is not clear, however, that the task of simulating a system is always simpler when using a simulation language.

It is obvious that as a computer language becomes specialized, it also becomes restrictive. It is also possible that an increase in specialization for a computer language will tend to decrease the efficiency of the language in terms of running time and allocated memory space. Motivated by these thoughts, the objective of this thesis is to compare the usefulness of two very common programming languages: GPSS and FORTRAN. GPSS is a specialized simulation language for the simulation of queueing problems and FORTRAN is a general purpose language. A detailed description of these languages is not contained in this thesis but may be found in Refs. 3, 4, 6, 7, 8, and 14.

The comparison of GPSS and FORTRAN in this thesis is made by simulating an identical queueing system in each of these languages. For the queueing system simulated, identical assumptions were made relative to modeling the system and the same exogenous data was used. Models in each of the languages provided comparable output data. At the beginning of this study, the author was equally familair with each language. Based upon these two simulations, the languages were then relatively evaluated according to the following criteria:

1. Ability to represent system
2. Simulation time
3. Ability to represent stocastic phenomena
4. Programming time
5. Computer running time
6. Data initialization and starting conditions
7. Storage requirement
8. Replication
9. Monitoring and debugging
10. Data collection and display

These criteria are discussed in Chapter II in light of their importance to simulation model building.

It should be made clear that the content of this thesis is directly related to a comparison of GPSS and FORTRAN in programming a simple queueing situation. No attempt is made to compare these languages beyond their application in programming this sample problem.

II. LANGUAGE COMPARISON CRITERIA

The criteria used to compare FORTRAN and GPSS represent both the economic considerations of choosing a language as well as the desirable features of a simulation language. The economic considerations [Ref. 12, pgs. 240-241] relate to cost and availability. Cost can be equated to the man hours required to obtain a validated computer model, together with the computer operating time required to obtain the final results. Availability includes the availability of computer hardware as well as the availability of knowledgeable programmers and technicians. The desirable features of a simulation language [Ref. 9, pgs. 26-38] are qualities that assist the programmer in conceptualizing, programming, debugging and experimenting with the computer model. A detailed explanation of the specific criteria utilized to compare FORTRAN and GPSS follows.

A. ABILITY TO REPRESENT SYSTEMS

Most systems of interacting entities can be said to possess two structures: static and dynamic. The static structure is independent of time, and consists of the framework or structure within which action takes place. The dynamic structure is time dependent and includes the processes and actions that occur within the static structure of the model.

In order to simulate a system, it is, therefore, important that the language used must contain an inherent framework or flexibility so that both the dynamic and static structure of the system can be easily and realistically included in the simulation. As examples of the two structures,

consider the operation of a message center. The transmitters, receivers, and electrical circuits represent the static structure. The arrival and departure of messages over the circuits make up the dynamic structure for this system.

B. SIMULATION TIME

In order to simulate the dynamics of a system, a method is needed which portrays the passage of time. Time in a simulation is controlled by means of a computer clock which is a programming mechanism that approximates the continuous flow of time in the simulation. Clocks are of two basic types: fixed time and next event. Fixed time clocks operate by advancing simulated time in discrete time intervals. At the completion of each time interval, a control mechanism scans all the possible actions that may take place and all computation scheduled for this interval is performed. The clock then advances into the next interval and the scanning process is repeated. Next event clocks utilize a control device that, at the completion of an event or computation, advances simulated time directly to the time of the next event scheduled to occur. For a detailed description of these clock mechanisms see Naylor [Ref. 12].

C. ABILITY TO REPRESENT STOCASTIC PHENOMENA

Events in the real world usually do not occur with regularity or according to some fixed pattern. The variability and uncertainty with which events occur are most often described by means of probability statements and distributions, e.g., event A has fifty percent chance of

occurring, or event B will occur every ten days plus or minus three days. To include these probability statements and distributions, a method is needed within the dynamic structure of the model to generate random numbers. Random numbers used in conjunction with probability distributions and statements are used to determine the outcome of stocastic events, e.g., did event A occur or not? In order to adequately model real world systems, it is therefore necessary to include in the evaluation of any computer language the ease with which the variability and uncertainty of events can be included in the simulation.

D. PROGRAMMING TIME AND COMPUTER RUNNING TIME

For any simulation, programming time and computer running time are important considerations. Both times represent dollars and, therefore, must be considered prior to choosing a computer language. Complex languages which compile and execute slowly in comparison to simpler languages may greatly reduce programming time due to their syntax. In deciding on a language to use, a cost and time trade off analysis should be made. This analysis can quite often only be made empirically, i.e., by comparing the results of the different languages as is being done in this thesis.

E. MONITORING AND DEBUGGING

An essential requirement for any programming language is that it ought to provide features that assist the programmer in debugging syntax and logic errors. For simulation languages in general, three desirable

debugging features are:

1. A list of compilation and execution errors referenced by statement number and clock time.
2. A list of execution errors reported with a complete printout of the status of the program at the time of error. This printout should contain as a minimum: clock time, status of appropriate variables and related parameters from the subroutines currently in effect.
3. A trace back feature which would provide some method to trace a given event thru the system.

These three features are also necessary in order to monitor the system dynamics. Without the ability to trace an event and the ability to observe the system at various times, many logic errors would be impossible to locate [Ref. 10, pgs. 35-36].

F. DATA INITIALIZATION AND STARTING CONDITIONS

Simulation studies often use extensive exogenous data. Therefore, convenient methods to initialize exogenous data into the computer model is a desirable language feature. Once the model has been initialized, the immediate objective then becomes to obtain information about the system at various points in time. In the analysis of some systems, the analyst may decide to study the system during steady state conditions, while for other systems, the transient conditions may be more important. Another desirable feature for comparison of language is then the ease

with which the inherent structure of the language allows the analyst to drive the simulation to a steady state condition. This must include, of course, a convenient method for minimizing the amount of execution time required to arrive at a steady state condition.

G. STORAGE REQUIREMENTS

Computer core storage is fixed for each model computer. This restricts the capability of each model computer to perform jobs within its memory capacity. In order to obtain the most efficient use of a computer's memory capacity, many computer facilities encourage programs to be written with small storage and time requirements by providing quicker turn around times to these programs. The language storage requirement therefore becomes important in obtaining shorter turn around times during the debugging and experimentation phase of the simulation. Core storage requirements of a computer language can therefore materially affect the time required to project completion.

H. REPLICATION

When simulation results are put to the test of statistical analysis, it is of primary interest to obtain results with a specified degree of confidence. In order to be able to specify a degree of confidence, a number of observations of the experiment must be obtained. These observations are obtained by replicating the computer experiment the desired number of times. The ease with which program replications can be made is therefore a measure of contrasting two simulation languages.

I. DATA COLLECTION AND DISPLAY

Data collection is a necessary requirement for any simulation. Ideally, data collection should be automatic, but in any event, data collection statements should not interfere with the operations of the model. Since completely automatic data collection is not practical, the next best alternative is to have certain information automatically collected and the remaining information available at the discretion of the programmer. The following data should be available as output at little or no inconvenience to the programmer [Ref. 10, pgs. 32-34].

1. The number of observations and the maxima and minima for all variables
2. Sums and sums of squares for time independent variables
3. Time-weighted sums and sums of squares for time-independent variables
4. Variable value histograms for time-independent variables
5. Time-in-state histograms for time-dependent variables
6. Time series plots over specified time intervals

The recording of results is a primary objective in every study. Therefore, the language selected for simulation should allow for varying output formats that are dependent upon the programmers needs. However, as is the case with data collection statements, the programmer should not have to spend a great deal of time in writing or debugging output data statements.

III. THE MODEL

This chapter introduces the torn tape relay network concept of operation in Section A and describes the system that was simulated for this case study in Section B. The FORTRAN and GPSS models of the torn tape relay network are described in Sections C and D respectively. The FORTRAN and GPSS programs are discussed with the intent of providing the reader insight into the methods that can be used by a programmer to simulate the static and dynamic structure of a system.

A. TORN TAPE RELAY CONCEPT

The purpose of a torn tape relay network is to move message traffic from an origin to a destination within the network. A typical torn tape relay system consists of a number of terminal and relay stations connected by circuits. A terminal originates and terminates messages for a military headquarters. Messages originated at a terminal are translated into a coded perforated tape. The perforated tape code is then transmitted to a relay station according to the assigned precedence and desired routing of the message. A relay station is typically connected to many terminal stations in the geographical area as well as to other relay stations. The mission of a relay station is to accept messages from the supported terminals and connecting relays and to retransmit the messages to the next station according to the messages routine instructions. This next station may be another relay or a supported terminal station.

A message may be assigned one of four precedences. Listed in order of importance from high to low, these precedences are: flash, immediate, priority, and routine. All messages are transmitted from one station to another on a first in first out, FIFO, discipline ordered by precedence. When a flash message is received at a station and no circuit is free to the next station in accordance with the routing instructions, a lower precedence message occupying a transmitting circuit is pre-empted. The flash message is then transmitted over the pre-empted circuit and the pre-empted message is refiled according to its original receipt time. The pre-empted message is then eventually retransmitted according to the normal FIFO discipline. A message with a flash precedence is the only type of message with pre-empting authority.

Prior to reaching the final destination, messages often remain in queue at some station and are said to be backlogged. Station backlogs are computed to be the time required to transmit all messages in queue for a given station. The total backlog is often separated into two categories: a high precedence message backlog consisting of flash and immediate messages and a low precedence backlog consisting of routine and priority messages. Backlog statistics are used as a measure of effectiveness to evaluate a station's ability to pass traffic.

The number of circuits interconnecting two stations regulates the backlog that exists between these stations. Additional tape transmitters, receivers, and personnel to operate the equipment are also needed as the number of circuits increase. The additional circuits, equipment and

personnel can be equated to dollars. Hence, in designing or operating a torn tape relay, one attempts to minimize the cost of operation by minimizing the number of interconnecting circuits, subject to some fixed effectiveness criterion. The maximum backlog that will be tolerated is often used as the fixed effectiveness criterion.

B. SYSTEM MODELED

The torn tape relay network that was modeled for this study consisted of five terminal stations and one relay station. A diagram of the system simulated is depicted in Figure 1. The terminal stations are numbered 1, 2, 3, 4 and 5. The relay consisted of five bays and are labeled 6, 7, 8, 9 and 10 in the diagram. Each bay contained relay equipment used to transmit and receive messages to a given terminal. The circuits interconnecting the terminals and relay bays are represented by directed lines in the diagram. Transmitting circuits are indicated by the direction of the arrow.

The five terminal stations and five relay transmit bays together with the associated circuitry comprised the static structure of the system. The dynamic structure of the system consists of messages possessing a given precedence, arrival time, message length and destination moving thru the static structure.

The problem addressed in this study was to simulate the torn tape relay network represented in Figure 1 and to determine the minimum number of circuits required between stations consistent with acceptable

TORN TAPE RELAY NETWORK

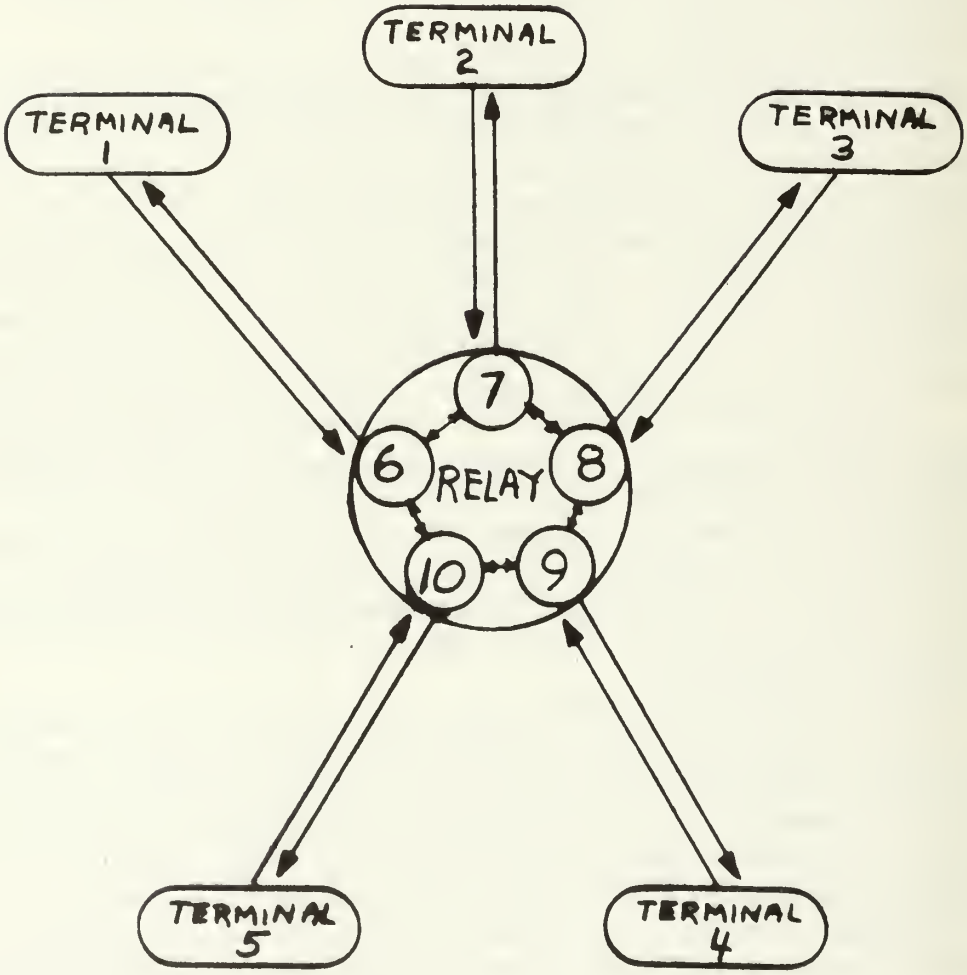


Figure 1

backlog figures. Backlogs were considered excessive if low precedence messages were backlogged in excess of 15 minutes and high precedence messages in excess of 2 minutes.

Various operating characteristics for all the stations were assumed for the system. These characteristics included probability distributions, message density functions, peak load figures and average message length. The following assumptions were used in modeling the torn tape relay network:

1. All messages introduced into the system reached the desired destination and no messages were misrouted.
2. Message center record keeping procedures did not contribute to backlog figures.
3 Circuit outages were not considered; hence, all circuits experienced one hundred percent reliability.
4. The time between message arrivals for each precedence message was distributed exponentially with mean peak hour values as listed in Figure 11, Appendix A.
5. The peak hour arrival rate was dependent upon time of day and was different for each site. The peak hour arrival rates utilized are listed in Figures 5 thru 9, Appendix A.
6. The average message length was not time dependent, but was dependent upon the station and precedence of the message. Message length was assumed exponential with mean values as listed in Figure 12, Appendix A.

C. FORTRAN SYSTEM MODEL

FORTRAN is a general purpose language, GPL, in that the language was not designed for special purpose applications such as simulation. FORTRAN is a comparatively simple language to learn, because the symbols and mnemonics are analogous to those of mathematics. For this reason, and due to the FORTRAN compilers universal availability, the language became a common computer language for scientific computations.

FORTRAN makes use of variables, subscripted variables, constants, expressions and functions. The number of different functions or subroutines used by a programmer is limited only by the size of the computer available. For simulation programming using FORTRAN, the programmer has great flexibility by being able to write a general main program containing a timing clock, and then calling various subroutines for execution.

In order to model the static structure of a system in FORTRAN, one must portray the structure of the system using the subscripted-unsubscripted variables and constants in conjunction with FORTRAN functions and expressions. The dynamic structure of a system is represented in FORTRAN by altering the flow of statement execution in a manner that reflects the action occurring in the actual system. The order of statement execution is varied in FORTRAN by use of IF statements, COMPUTED GO TO statements and logical type statements.

In order to simulate the torn tape relay network in FORTRAN, the static structure was represented by subscripted arrays. Four three dimensional arrays, namely RR (i,j,k), PP (i,j,k), OO (i,j,k) and

FLASH PRECEDENCE ARRAY FF(i,j,k)

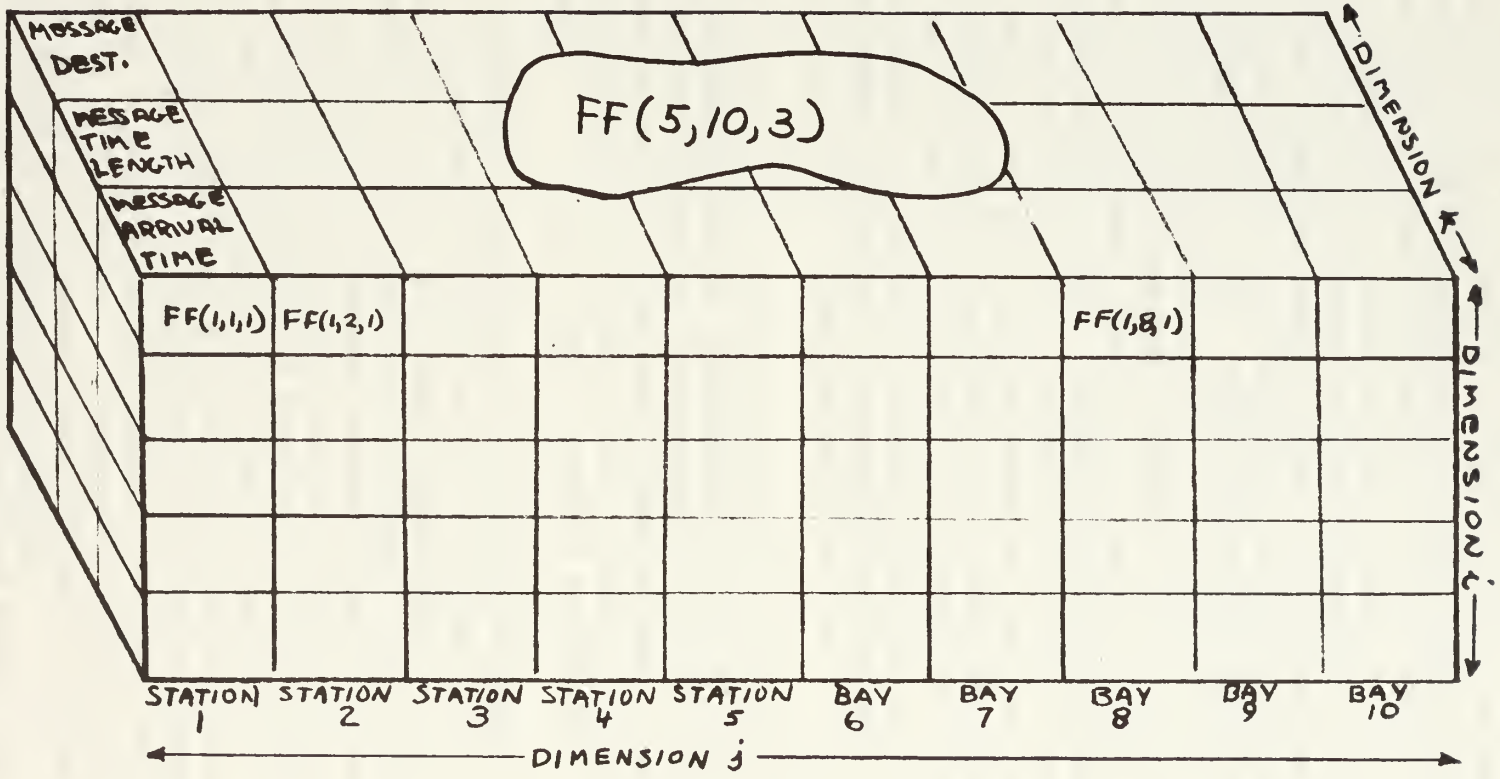


Figure 2

FF (i,j,k), were used to store information pertaining to routine, priority, immediate and flash messages, respectively. Figure 2 contains a representation of the FF array. The k dimension of each array was equal to three, and was used to store the arrival time, required transmission time, and destination of each message waiting for transmission at the terminals and relay transmit bays. The i and j dimension of these arrays fixed respectively the number of messages and the station at which the messages were currently located. The j dimension was equal to ten with argument one thru five corresponding to the five terminals and arguments six thru ten corresponding to the five relay transmit bays. The arguments of the j dimension corresponded to the station numbers depicted in Figure 1.

The d eimension was different for each precedence array and was selected to permit at least a thirty minute backlog of messages to exist for each precedence category at each station. For k equal one, message arrival times were stored in the i dimension in increasing order.

Ten one dimensional arrays HOLD1(i) thru HOLD10(i) were used to represent the circuits between terminals and relay bays. The arrays HOLD1 thru HOLD10 correspond to the numbering of Figure 1; these arrays contained the time at which a specified circuit would be free. The number of transmit circuits from one location to another was equal to the dimension of the HOLD array corresponding to this pair of stations.

The dynamic structure of the system was represented by moving message parameters from the terminal storage areas represented by

position 1 thru 5 of the j dimension to the desired relay transmit bay represented by position 6 thru 10 and then from the relay to the desired destination. The order of statement execution was controlled by a FORTRAN COMPUTED GO TO statement that acted as a next event clock. Program control was transferred to the station that had the highest precedence message with the lowest arrival time available for transmission. A message was available for transmission if the scheduled message arrival time was less than or equal to the simulation clock time and there was a free circuit in order to transmit the message to the connecting station. Transmission from a terminal was accomplished by transferring message parameters from the array column representing the terminal to the array column representing the relay transmit bay that will transmit the message to its final destination. In addition, the message transmission time is added to current simulation time and this value is placed in the proper HOLD array.

For example, suppose a flash message from terminal two to terminal three was selected as the next event in the simulation. Since relay transmit bay eight transmits to terminal three, the message parameters from column two would be transferred to column 8 of array FF. The arrival time of the message for terminal eight would also be updated by the transmission time of the message. In addition, the time at which transmission ends would be transferred to HOLD2(i).

The information was deleted from the HOLD array after the required message lapse time by use of the next event clock. Each time a set of

message parameters was transferred from a terminal column, the remaining message parameters moved forward one position, and a new set of message parameters was created to occupy the empty position. After the creation of a new message, all positions in the j dimension column were filled, and the message arrival times were in ascending order. Once a message was transferred to a relay transmit bay column, the message arrival times in the column were not necessarily in ascending order. Hence, after a message was transferred to a relay transmit bay column, all message parameters in the column were ordered to insure the lowest arrival time message was in position one and therefore was the next scheduled message to be transferred from the column.

Messages in a relay transmit bay column selected as the next event were processed in a similar manner. The single exception being that messages transmitted by the relay were destined for the final destination. Hence, after a message was held in the respective HOLD array for the messages transmit time, the message attributes were destroyed. The destruction of the message's parameters represented the successful delivery of the message.

Backlogs were computed after transferring a message to the next station. A message was backlogged if simulation time was greater than the message arrival time stored in the j dimension of the respective precedence array. All messages stored in relay terminal bay columns were backlogged, whereas, only the messages with arrival times less than clock time were backlogged in the terminal columns. The messages

FORTRAN FLOW CHART

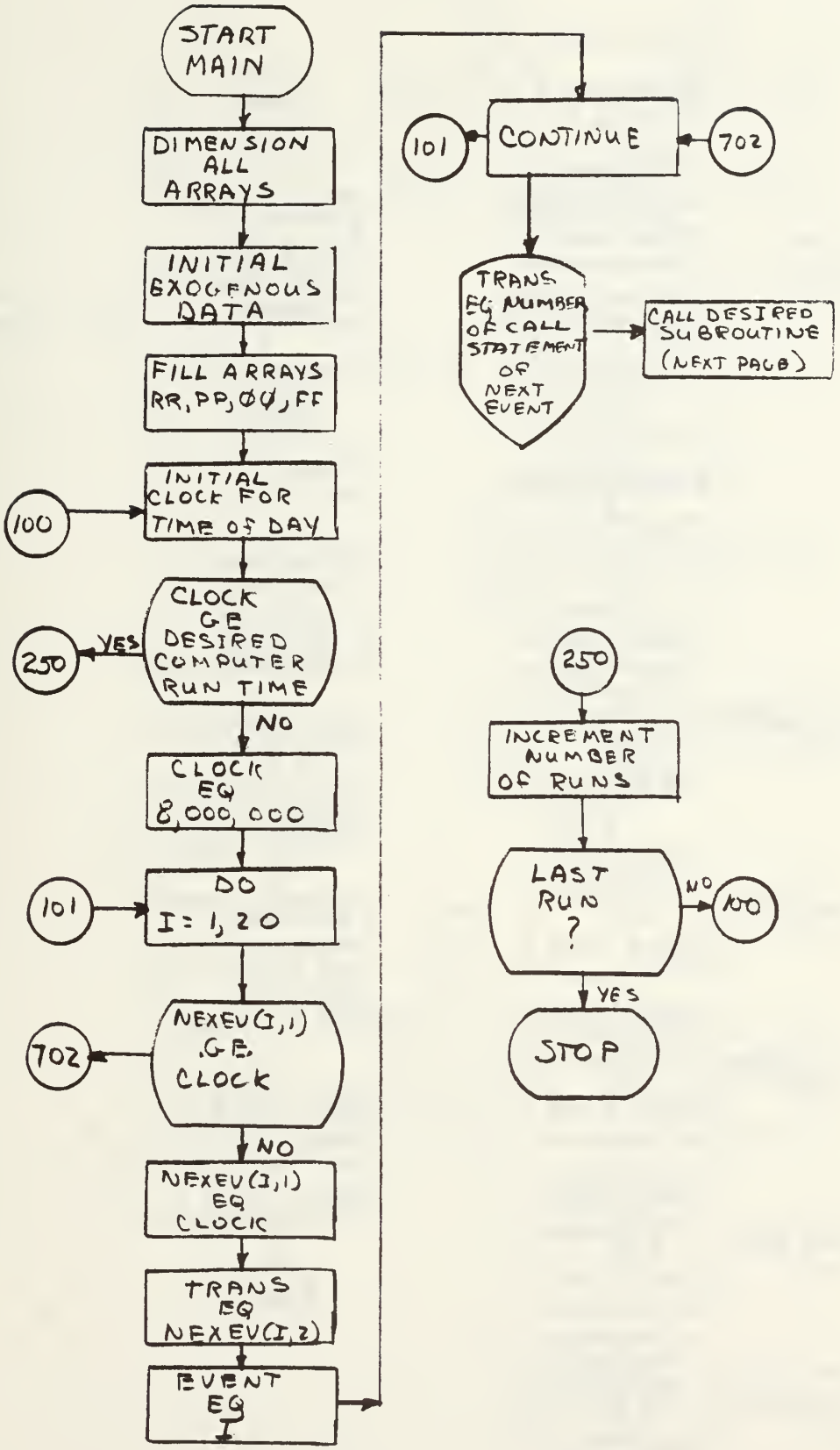
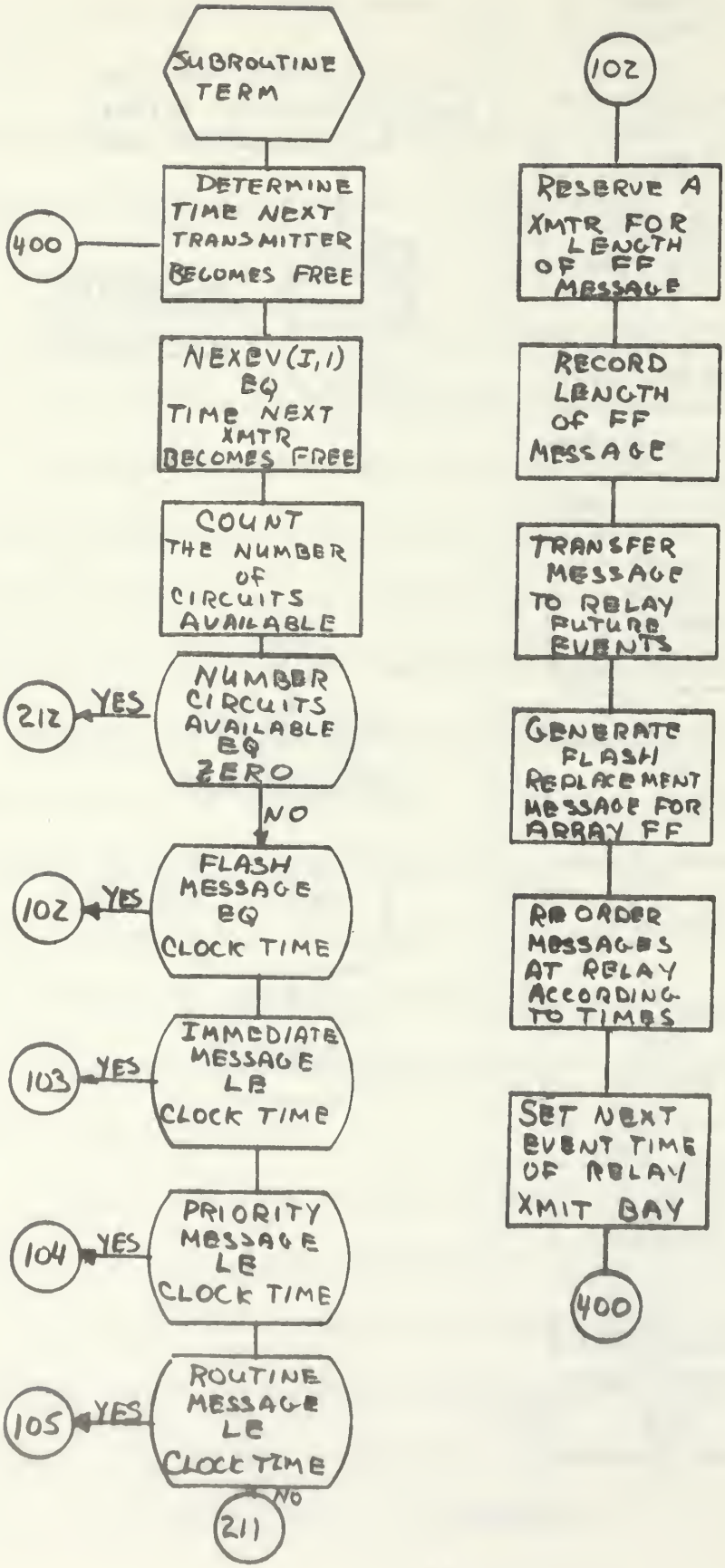
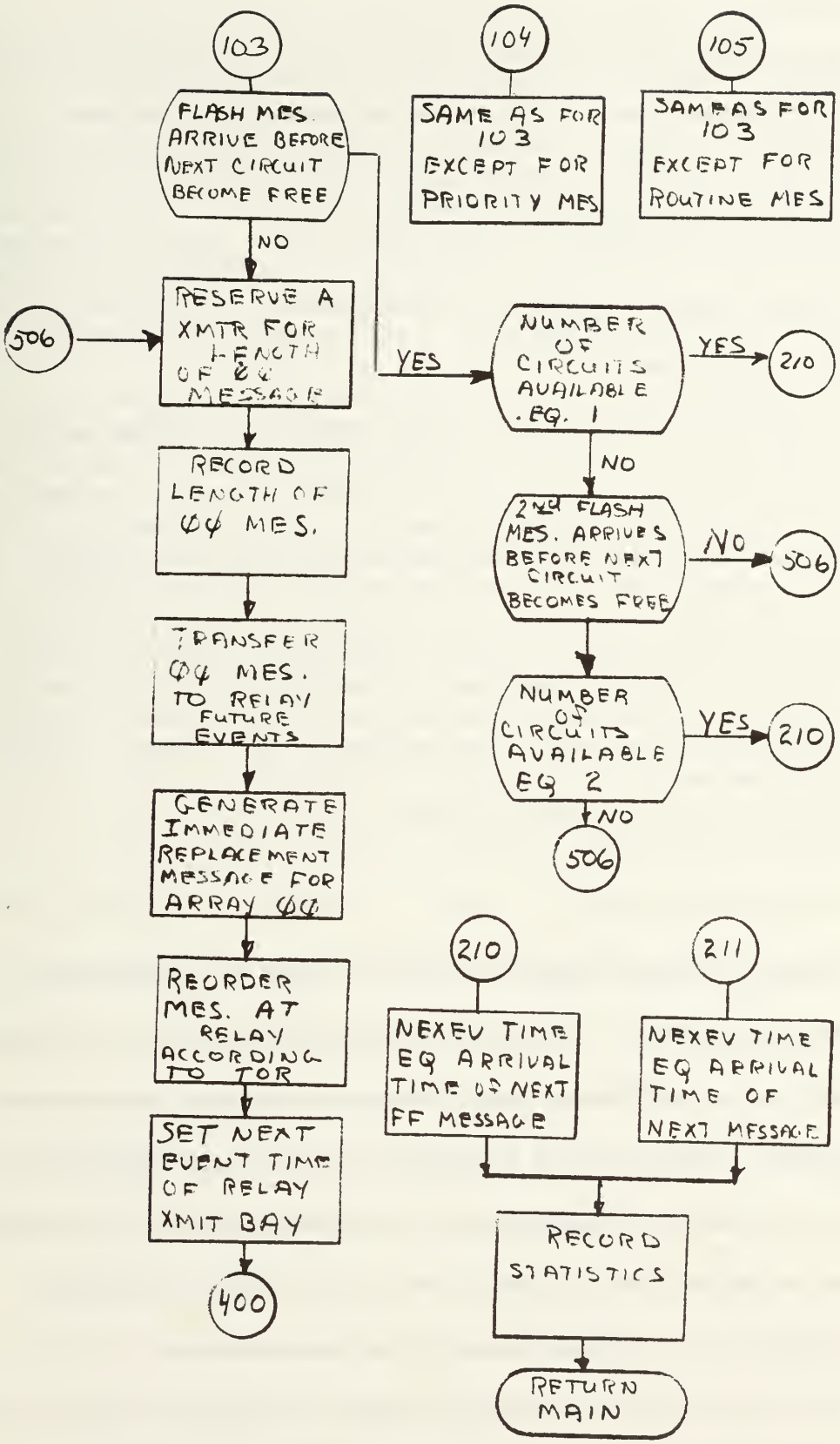


Figure 3


SUBROUTINE TERM
400
DETERMINE TIME NEXT TRANSMITTER BECOMES FREE
NEXEV(I,1) EQ TIME NEXT XMTR BECOMES FREE
COUNT THE NUMBER OF CIRCUITS AVAILABLE
212
YES
NUMBER CIRCUITS AVAILABLE EQ ZERO
NO
102
YES
FLASH MESSAGE EQ CLOCK TIME
103
YES
IMMEDIATE MESSAGE LE CLOCK TIME
104
YES
PRIORITY MESSAGE LE CLOCK TIME
105
YES
ROUTINE MESSAGE LE CLOCK TIME
NO
211
102
RESERVE A XMTR FOR LENGTH OF FF MESSAGE
RECORD LENGTH OF FF MESSAGE
TRANSFER MESSAGE TO RELAY FUTURE EVENTS
GENERATE FLASH REPLACEMENT MESSAGE FOR ARRAY FF
REORDER MESSAGES AT RELAY ACCORDING TO TIMES
SET NEXT EVENT TIME OF RELAY XMIT BAY
400


103
FLASH MES. ARRIVE BEFORE NEXT CIRCUIT BECOME FREE
NO
YES
104
SAME AS FOR 103 EXCEPT FOR PRIORITY MES
105
SAME AS FOR 103 EXCEPT FOR ROUTINE MES
506
RESERVE A XMTR FOR LENGTH OF ØØ MESSAGE
RECORD LENGTH OF ØØ MES.
TRANSFER ØØ MES. TO RELAY FUTURE EVENTS
GENERATE IMMEDIATE REPLACEMENT MESSAGE FOR ARRAY ØØ
REORDER MES. AT RELAY ACCORDING TO TOR
SET NEXT EVENT TIME OF RELAY XMIT BAY
400
NUMBER OF CIRCUITS AVAILABLE .EQ. 1
YES
210
NO
2ND FLASH MES. ARRIVES BEFORE NEXT CIRCUIT BECOMES FREE
NO
506
NUMBER OF CIRCUITS AVAILABLE EQ 2
YES
210
NO
506
210
NEXEV TIME EQ ARRIVAL TIME OF NEXT FF MESSAGE
211
NEXEV TIME EQ ARRIVAL TIME OF NEXT MESSAGE
RECORD STATISTICS
RETURN MAIN

with times greater than the clock time in the relay columns were future events and had not yet been introduced to the model. The backlog for a station was computed by totaling message time lengths for those messages with arrival time less than or equal to clock time. The maximum backlog that occurred during a computer run was recorded for each terminal station and relay transmit bay.

A flow chart of the torn tape relay network in FORTRAN logic is illustrated in Figure 3. This is an abbreviated flow chart and includes only the main program and the programmed operation of a terminal. Although some detail is omitted, Figure 3 portrays a system flow chart for FORTRAN. A complete flow chart of the FORTRAN model is included in Appendix B, together with an explanation of the symbols, and description of the arrays and subprograms used.

D. GPSS SYSTEM MODEL

General Purpose Simulation System, GPSS, is a popular simulation programming language. GPSS was designed to be used as a simulation language and as such makes special features available to simulation programmers. These features hopefully reduce the time required to simulate a system by reducing problem formulation time, programming time and debugging time. GPSS was designed by the International Business Machines Corporation and is well documented.

GPSS uses a block diagram flow chart procedure to represent the static structure of a system. The blocks used to represent the static structure are divided into six categories. The most fundamental of

these categories contain the GPSS entities STORAGES, FACILITIES and LOGIC SWITCHES. These entities are the building blocks of GPSS. A FACILITY allows entry to only one transaction at a time. A transaction is a dynamic entity and is used to represent movement in the system. The length of time a FACILITY is occupied by a transaction may be determined by a transaction's parameter value. A parameter describes a particular transaction attribute. STORAGES allow entry to several transactions simultaneously. LOGIC SWITCHES are two position gates which alter transaction flow according to the status of some other entity or transaction parameter value. The dynamic structure of a system is modeled in GPSS by the use of transactions. Transactions are created and destroyed as needed during the computer simulation. Transactions may have associated with them a set of parameters. The values assigned to these parameters remain with the transaction until they are changed or the transaction is deleted from the model.

Associated with GPSS entities are certain standard numerical attributes. The value of these attributes may be referred to or collected by the programmer during the execution of the program. Standard numerical attribute values may be used to trigger LOGIC SWITCHES, assigned to transaction parameters, as well as stored for future use.

For the torn tape relay network, the terminal and relay sites, transmitters, receivers, and interconnecting circuits were modeled by use of a block diagram. The messages were represented by transactions which moved through the block diagram according to the instructions of

the model. Transactions were created in the model for each precedence, and from each terminal according to an exponential distribution. The intermediate and final destinations, precedence, and message length were assigned to transaction parameters in the torn tape relay model.

Extensive use was made of the FACILITY entity. FACILITIES were used to represent torn tape transmitters. Transactions waited in and advanced in queue according to the FIFO discipline. A transaction occupied a facility, for the simulation time length of the message. A PREEMPT block allowed transactions representing flash messages to pre-empt a FACILITY occupied by a lower precedence transactions.

Entities may be referred to by indirect addressing in the GPSS program. Normally, each entity is designated by number or mnemonic in the flow chart of the system. However, it is also possible to designate a particular entity by the value of a transaction parameter. Indirect addressing is often used to specify entity numbers when there are several identical processes occurring in the system to be modeled.

In the torn tape relay model, five terminals and five relay transmitter bays performed identical operations. Each location transmitted messages according to the FIFO discipline. The terminals transmitted to the relay, and each relay transmitter bay transmitted messages to a particular terminal. Therefore, since the functions at each location were identical, the required programming was reduced tenfold by incorporating indirect routing for referencing most entity blocks.

A flow chart in GPSS of a single terminal is depicted in Figure 4. This flow chart differs slightly from the actual GPSS model used in that indirect routing, statistic gathering and storage of transactions in user chains is omitted. A complete flow chart of the model is included in Appendix C together with an explanation of symbols used. A comprehensive discussion on GPSS may be found in IBM manuals and Schriber [Refs. 6-8,14].

GPSS FLOW CHART

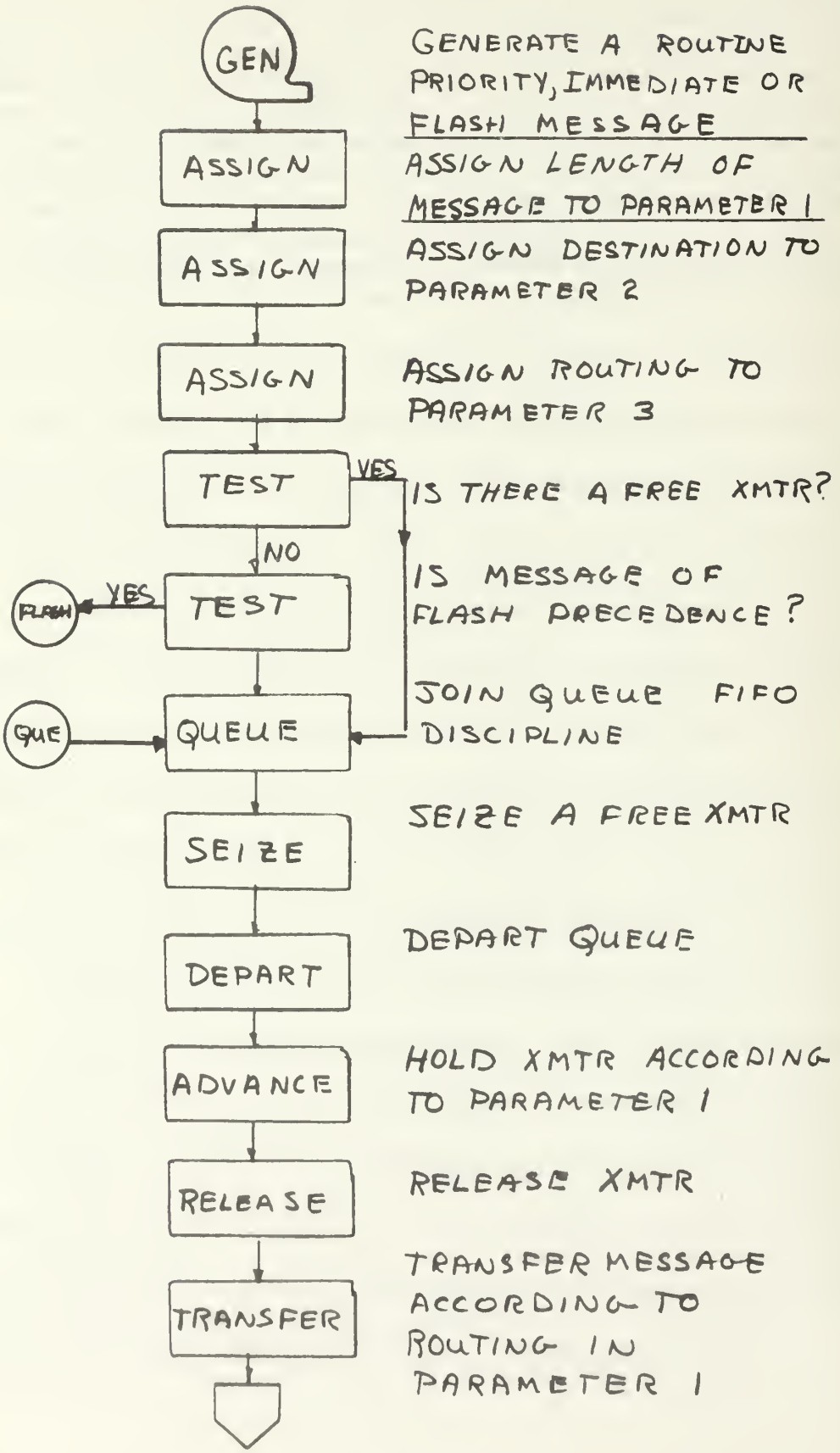
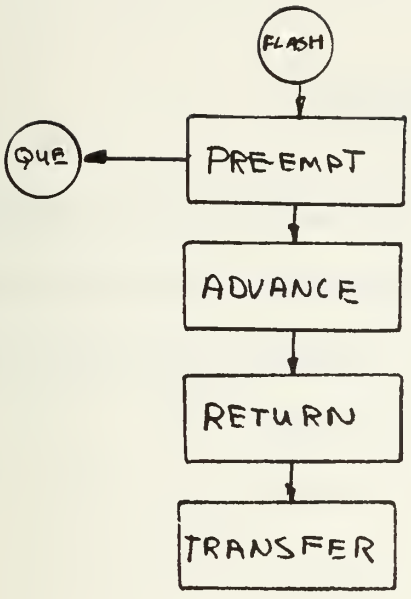


Figure 4

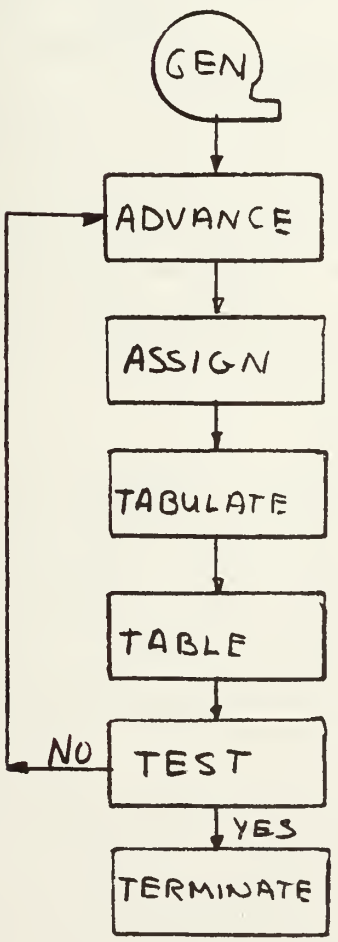


FLASH MESSAGE PREEMPTS LOWEST PRECEDENCE MESSAGE

HOLD XMTR ACCORDING TO PARAMETER 1

RETURN XMTR TO NORMAL MES. TRAFFIC

TRANSFER MESSAGE ACCORDING TO ROUTING IN PARAMETER 2.



GENERATE 1 TRANSACTION

HOLD TRANSACTION FOR LENGTH OF A COMPUTER RUN

ASSIGN STATISTICS TO PARAMETERS 1-30

RECORD PARAMETER VALUES IN TABLES

ASSIGN FREQUENCY TABLE RANGES

DESIRED NUMBER OF COMPUTER RUNS?

DESTROY TRANSACTION

IV. GPSS-FORTRAN COMPARISON

GPSS and FORTRAN are evaluated in this chapter using the criteria discussed in Chapter II. The evaluation is based upon observations and experiences encountered while modeling the torn tape relay network discussed in Chapter III.

A. ABILITY TO REPRESENT SYSTEM

1. FORTRAN

The torn tape relay network was first simulated in GPSS and then in FORTRAN. The author was hence quite familair with the system prior to programming the system in FORTRAN. Even so, a considerable amount of time was consumed trying to conceptualize the problem in the form of subscripted-unsubscripted variables. Many methods proved infeasible due to the requirement to record backlog statistics and due to the pre-empt authority of flash messages.

In order to collect backlog statictics, backlogged messages were simulated by creating messages in advance with a scheduled time of arrival. When simulation time advanced and equaled the arrival time of a message, if a circuit was available to the desired destination then the message was transmitted. If a circuit was not free, the message would wait in queue and be processed according to the FIFO discipline. The only practical method to store messages waiting for an available circuit, appeared to be to store the message parameters in arrays. Since

arrays were used, the following decisions had to be made before a system static structure could be formulated:

1. How should message parameter values be assigned to arrays?
2. How many arrays and what dimensionality should be used?
3. What properties for each message should be retained?
4. How is information to be exchanged between arrays in order to represent the dynamic structure of the system?

2. GPSS

The torn tape relay network was comparatively simple to represent in GPSS. The system static structure was logically represented by means of a GPSS block diagram similar to Figure 4 of Chapter III. The blocks FACILITY, STORAGE, TRANSFER, QUEUE and PREEMPT readily portrayed the realistic operation of the system. Once the entire block diagram was completed, the programming of the model in GPSS followed directly, since there exists a one to one correspondence between GPSS blocks and GPSS program statements. The dynamic structure of the torn tape relay problem was represented in GPSS by GPSS transactions. For this problem, transactions were considered messages. Transactions were created at prescribed interarrival rates and assigned parameter values corresponding to precedence, message length and destination. The flow of transactions followed the block diagram structure of the network.

3. Comparison

The torn tape relay network system was much easier to represent in GPSS than in FORTRAN. The construction of the GPSS block diagram using the block mnemonics facilitated building a GPSS model of the system. Initially the system was difficult to conceptualize using FORTRAN statements since it was possible to become involved in the details of FORTRAN and lose presence of the system to be modeled. The flexibility of modeling a system in FORTRAN can be a handicap for less experienced programmers because a programmer must select from among apparent alternate methods of structuring the model. Hence, it can be difficult and time consuming to determine which methods are feasible and which feasible method is best.

B. SIMULATION TIME

1. FORTRAN

The incremental time clock and the next event clock were considered for use in the FORTRAN model. The incremental time clock was not used, however; this clock appeared to be simple to utilize but did not appear very efficient since simulation time would be advanced in increments of one second and many hours of operation had to be simulated. The next event clock was used and appeared to be more efficient since simulation time is advanced directly to the time of the next event.

In retrospect, the incremental clock might have been a more efficient choice, since time progressed in small time increments in the

model. If an incremental clock had been used, FORTRAN bolean variables could have been incorporated in lieu of the FORTRAN IF statements and the COMPUTED GO TO statement. Bolean variables associated with each possible event could have been set to a go condition if the event was scheduled to occur during the next time increment. Bolean variables require considerably less computer time and hence might have resulted in a more efficient time clock.

2. GPSS

The timing mechanism of GPSS was a next event clock. The GPSS program operated by moving transactions thru a block structure that represented the static structure of the system. Each block represented the possible occurrence of an event in the system. The GPSS master program maintained a record of when these events were scheduled to occur and processed the events in their proper sequence in time.

3. Comparison

The most important difference in comparing GPSS and FORTRAN with respect to simulation time is that GPSS has a built in timing clock and FORTRAN does not. When using GPSS the simulation of a system must include the next event clock. However, no programming or structuring of this clock mechanism is required since the clock feature of GPSS is entirely automatic within the block structure. The simulation of a system in FORTRAN does require the programming and structuring of a clock mechanism to meet the specific needs of the system being simulated.

C. ABILITY TO REPRESENT STOCASTIC PHENOMENA

1. FORTRAN

The mathematical structure of FORTRAN facilitated the programming of probability distributions, functions and functional relationships. Many useful mathematical functions are available directly from the FORTRAN library. In addition, subroutines are available to generate random numbers and variates from most of the ordinary theoretical probability distributions. Variates can also be easily generated from empirical distributions, see Naylor [Ref. 12].

Random numbers were generated in the FORTRAN torn tape relay model using a library subroutine. The inverse transformation technique was then used to obtain variates from an exponential probability distribution which required the evaluation of a natural logarithm. Several methods were available to generate random numbers and exponential variates, some of which were more efficient than those used. The methods used were selected due to their simplicity and immediate availability.

2. GPSS

GPSS contains eight independent random number generators. These generators can be synchronized or can operate independently at the discretion of the programmer.

Two methods are available to generate variates from probability distributions other than uniform. One method uses the GPSS FUNCTION statement and a CDF approximated by linear segments. This

method generates variates by using a random number argument and linear interpolation on the CDF approximation. This method was used to generate exponential variates for the torn tape relay model.

The second method uses a GPSS VARIABLE statement to express the CDF in closed form. Variates can then be generated using the CDF's inverse transformation function or some other acceptable method. This method is inconvenient, however, since GPSS contains no internal routines to evaluate functions such as natural logarithm, sine, gamma, squareroot, etc. These functions must be evaluated by using GPSS VARIABLE statements to include the evaluation of power series, etc.

3. Comparison

GPSS provides adequate methods to generate variates from a probability distribution in order to represent stocastic phenomena. The presence of eight random number generators was convenient, but the lack of GPSS routines to compute common mathematical functions could prove inconvenient. Stocastic phenomena are easily represented in FORTRAN and subroutines are available to generate random numbers. Any number of independent random number generators can be included in the FORTRAN program. Variates from probability distributions can be obtained by linear interpolation of the approximated CDF or from the theoretical distribution in both languages. However, the availability of FORTRAN library functions often simplifies computing closed form probability functions compared to the GPSS method.

D. PROGRAMMING TIME

1. FORTRAN

FORTRAN statements can be written with or without the aid of a flow chart. A general flow chart containing narrative descriptions of the actions that are to be portrayed in FORTRAN is often quite useful. Such a flow chart helps solidify model concepts and often suggests methods of approach. A detailed statement by statement flow chart is seldom required except for extremely complicated portions of a program. The writing of the FORTRAN program for the torn tape relay problem, after a narrative flow chart had been developed, was comparatively simple. The FORTRAN program required in excess of 600 statements to model the system.

2. GPSS

GPSS programming time was considered one of the major advantages of the language. The GPSS block diagram used to simulate the system greatly simplified the statement programming of the model. Once the block diagram had been created, it was a simple matter to write the program statement associated with each GPSS block. The GPSS program consisted of 200 statements of which fifty were FUNCTION and VARIABLE statements and sixty were used to collect statistics for output purposes.

3. Comparison

The programming phase of the simulation in GPSS was significantly faster than in FORTRAN. Often where one statement was

needed in GPSS to represent an event, a subroutine was required in FORTRAN to represent the same event. Consequently, the FORTRAN program required three times as many statements as needed by GPSS in order to perform the same task. The time required to program the FORTRAN model was significantly higher than the three to one ratio indicated by the number of statements required. It is quite conceivable that a GPSS programmer could be experimenting with the completed GPSS model while a FORTRAN programmer was still investigating a method of attack to program the same system.

E. COMPUTER RUNNING TIME

1. FORTRAN

The FORTRAN model required seventeen minutes to compile and execute the same number of runs conducted by the GPSS model in six minutes. One would generally expect that the FORTRAN program would execute faster than the GPSS program. This expectation is based on the fact that FORTRAN is a lower level language and utilizes less sophisticated data structures and methods to vary the flow of statement execution. An attempt was made to determine the reason for this lengthy execution time.

The probable cause of the excessive time was due to the method used to advance messages in arrays RR, PP, OO, FF. After a message was transferred from a terminal station array column, all message attributes were moved forward one position prior to a new

message being created. After the message was transferred to a relay transit bay column, all message arrival times were ordered to insure the message with the smallest arrival time was in position one of its respectiv column. The transmission of one routine message required changing several hundred computer storage values. This perpetual changing of storage values undoubtedly resulted in a high FORTRAN program execution time.

2. GPSS

The GPSS model of the torn tape relay network required six minutes and five seconds to execute eight repetitions of the torn tape networks peak period of operation.

3. Comparison

Although it may be possible to write faster executing programs in FORTRAN than in GPSS, the torn tape relay example indicates that this is not automatic. It appears that the GPSS language includes efficient routines which when assembled by a programmer into a computer program provide a comparatively efficient method of building a model. A GPSS program may not be as efficient as theoretically possible in FORTRAN. However, for the inexperienced programmer, the efficiency of the GPSS language may be difficult to match.

The GPSS program written for this study was three times faster in computer execution time than the FORTRAN program. Twenty-five replications could have been obtained using the GPSS program as compared to eight using the FORTRAN program in the same amount of

computer time. These seventeen additional observations would have significantly increased the sample size and provided more confidence in the results obtained.

F. MONITORING AND DEBUGGING

1. FORTRAN

No set debugging scheme was available to ferret out logic errors in the FORTRAN program. The syntax diagnostics available are quite often very hard to correlate with program logic errors. As an example the FORTRAN program developed several undesired loops that held simulation time constant while exhausting computer execution time. The location of these errors was difficult to determine in a single diagnostic run since the simulation clock time of the errors had to be determined. This was done by printing the simulation time after each message was processed. The printout was then used to determine the simulation time when the undesired loop first occurred. A second diagnostic run was then used to obtain information dumps corresponding to the model simulation time encompassing the undesired loop. A dump consisted of the simulation time and the relevant arrays. By using this data, logic errors were isolated by tracing the movement of message parameters among array columns. This method was quite time consuming since there existed in excess of 3,000 relevant storage locations. On several occasions the error could only be isolated to a particular subroutine and additional runs and output were required to locate the error.

Once the location of the undesired loop was located, however, it was a comparatively simple task to determine the cause and correct the error.

2. GPSS

GPSS debugging and monitoring facilities were excellent. Chapter 17 of the IBM User's Manual [Ref. 7] provides the programmer with many valuable hints on monitoring and debugging the GPSS program. Errors detected during execution of the program triggered an elaborate information dump which automatically provided the following information:

1. A coded error message
2. Simulation time error was encountered
3. The block number a transaction was currently in
4. The next block a transaction was to be processed by
5. The number of transactions processed by each block in the program
6. Current and future event chains
7. GPSS FACILITY statistics
8. QUEUE statistics
9. STORAGE statistics

This dump included ample information to isolate the majority of the errors encountered debugging the torn tape relay model. However, on occasion, an additional monitoring and debugging feature was required. This feature was the GPSS TRACE option. The TRACE feature provided a capability of tracing a transaction with a given attribute thru the static structure of the model. Transaction location and parameter values were

printed as the transaction followed the GPSS block diagram. In addition, information dumps were easily obtained at strategic locations in the program by using a traced transaction to trigger the dump. The dynamics of the model were then reconstructed using both the information dumps and the trace block data. From this information, the cause of error was easily located.

3. Comparison

The GPSS automatic information dump that occurred when an error was detected facilitated the isolation of 90 percent of the errors encountered. When additional diagnostic information was needed, the GPSS TRACE option was available. These two features greatly simplified the task of monitoring the debugging the GPSS torn tape relay network program. In contrast, the FORTRAN debugging and monitoring error detection schemes were not suited for detecting logic errors in the simulation model. The method used by the FORTRAN programmer must depend on the type of error encountered and the particulars of the model. Error detection required ingenuity, time and often luck in order to locate the cause of errors in the FORTRAN torn tape relay model.

G. INPUT DATA AND INITIALIZATION

1. FORTRAN

Exogenous data for the torn tape relay was made available to the model conveniently by use of FORTRAN DATA statements. In particular it was found by experimentation that the model did not have to be preloaded with backlogged messages at the beginning of the first

run. Using DATA statements, the terminal stations and relay transmit bay stations corresponding to the columns of arrays RR, PP, OO, FF could have been pre-filled with a desired number of backlogged messages of any length, destination and arrival time.

2. GPSS

Exogenous data for the GPSS version of the torn tape relay network was initialized into the model by means of the GPSS FUNCTION. Although transactions were not preloaded into the GPSS version of the model to represent a backlogged message condition, preloading could have been easily accomplished using the GENERATE, and MATRIX blocks. In GPSS a programmer can specify an upper limit on the number of transactions that will be created by a particular GENERATE block and GENERATE blocks could have been used to create the desired number of backlogged messages for each precedence category and station.

3. Comparison

Exogenous data initialization was handled adequately by both GPSS and FORTRAN for the torn tape relay network system. Although the model was not preloaded with backlogged messages for start-up, it appeared that both GPSS and FORTRAN could have handled the requirement without difficulty.

Although DATA statements were used for exogenous data input for the FORTRAN model, additional methods were also available, e.g., data could also have been inputed by means of magnetic tape or punched cards. GPSS relies heavily on the FUNCTION statement,

SAVE VALUE, MATRIX and INITIAL blocks to handle exogenous data requirements. GPSS cannot accept magnetic tape input and can only read data cards with the assistance of the HELP block. However, the IBM manual [Ref. 7], states that the HELP block is intended only for users who are thoroughly familiar with the internal operation of the GPSS program. This last requirement tends to eliminate many hopeful simulators.

GPSS and FORTRAN adequately handled the exogenous data requirement for the torn tape relay network. However, the FORTRAN language would be preferable to GPSS if the system to be simulated required voluminous input data.

H. STORAGE REQUIREMENTS

1. FORTRAN

The FORTRAN model of torn tape relay network required 64K bytes of computer storage.

2. GPSS

The GPSS model of the torn tape relay network required 128K bytes of computer core storage. GPSS can be operated on the IBM/360/67 in either a 128K or 256K mode. The 256K mode enables a programmer to utilize an increased number of entities of each category. The 128K version allocates 150 STORAGE blocks, whereas the 256K version allocated 300 STORAGE blocks. The only critical factor is the total amount of storage required by a program. Unused space from one entity group can be reallocated to another entity group and this allows for an increase in the number of blocks allocated to a specified entity group.

An area referred to as GPSS COMMON also requires allocation of storage space. GPSS COMMON is the space used dynamically for transaction information, temporary data, and for storing statistical data. The GPSS torn tape relay network model required COMMON storage in excess of that allocated to the 128K version. The additional storage for the torn tape relay model was obtained from unused entity blocks.

3. Comparison

The FORTRAN model required half the storage required by the GPSS model, that is 64K versus 128K. This reduced storage requirement did not reflect a faster turn around time for the FORTRAN program due to the long execution time of the program.

I. REPLICATION

1. FORTRAN

Replication for the FORTRAN version of the torn tape relay network was accomplished by use of a FORTRAN IF statement in conjunction with the next event clock. Prior to selecting the next event, a test was conducted to determine if the current simulation time was in excess of the maximum simulation time for a run (9,000 seconds). If simulation time was in excess of 9,000 seconds, statement execution was transferred to record pertinent statistics. After the statistics were recorded, the number of replications completed was incremented and tested. If the required number of runs had been completed, the simulation was terminated, if not, the clock was reset and another replication was performed.

2. GPSS

Replication is accomplished in GPSS by use of RESET and CLEAR blocks and by the redefinition of blocks. The RESET block sets all statistical tables to zero except those specified by the programmer. The RESET block does not alter the status of any block entities or alter the values of any transaction parameter values. The CLEAR block performs a function similar to the RESET clock except that in addition all transactions currently in the system are destroyed.

The GPSS torn tape relay model was replicated with the aid of RESET blocks and the redefinition of entity blocks in conjunction with a specially constructed timing program. The timing program was a simple subprogram that controlled the simulation time for each replication. The program consisted of a GENERATE block that created a transaction. This transaction then waited in an ADVANCE block for the desired time length of each replication (9,000 seconds). The transaction was then used to gather and store desired model statistics in GPSS TABLES. After the statistics were gathered, the transaction was destroyed. A RESET block then caused entity block statistics to be zeroed except for specified tables. A START block was then redefined which caused a new transaction to be created and the above process was repeated.

3. Comparison

The GPSS features available to assist the programmer in replication are very powerful. Specific entity blocks were designed

to assist the programmer in every facet of replication. Undoubtedly, one could program the same features into a FORTRAN program, but to do so would be programmer time consuming and require programmer ingenuity.

J. DATA COLLECTION AND DISPLAY

1. FORTRAN

FORTRAN does not provide any output automatically and all output desired must be programmed for within the model. This may often require ingenuity and a considerable amount of programming time. Of importance,however, is that desired output can always be obtained in any format.

2. GPSS

The GPSS program stored statistics in GPSS SAVEVALUES during the execution of a replication, and the GPSS timing program was used to collect desired statistics after each replication. The statistics were stored until the completion of the last replication and then printed in tabular form.

Some output from GPSS is automatic while other output can be programmer specified. The automatic output consists of the following:

1. Block counts
2. Current value stored in SAVEVALUES
3. USER CHAIN statistics
4. QUEUE statistics
5. FACILITY statistics

6. STORAGE statistics

7. Relative and absolute clock times

The automatic output is voluminous and provides general information about the system.

Information relating to transaction,parameters time and specific facets of the system must be programmer specified. The output can be obtained either in frequency table or histogram form, the intervals for which are specified by the programmer. Mean values, standard deviations, maxima, minima and number of observations in each frequency class are provided as automatic output for table arguments. All attributes of a modeled system can conceivably be tabulated in a frequency table or plotted in a histogram. The format of a frequency table cannot be altered. The programmer can, however, program the dimensions of a histogram.

3. Comparison

Data collection and display in FORTRAN is flexible, obtrusive and programmer time consuming. In contrast, data collection in GPSS is non-flexible, unobtrusive and rapidly programmed. FORTRAN is flexible in that all data can be displayed according to any desired format. The GPSS format is compact and informative but it cannot be altered by the programmer.

Data collection and display is obtrusive in FORTRAN because data collection methods can interfere and obscure the simulation logic

of the system. GPSS data collection and display statements do not interfere with program logic and can normally be placed in a separate portion of the program to avoid obscuring the logic of statement execution.

FORTRAN data collection and display techniques require considerable programmer time in engineering the desired display format. Additionally, all desired output from FORTRAN programs must be programmed since no output is provided automatically. GPSS automatically provides a considerable amount of output and allows the programmer to collect other data at his discretion. The programmer specified output is rapidly programmed within the simulation since the output format does not have to be considered.

V. CONCLUSIONS

The key words in describing the usefulness of FORTRAN for simulation is flexibility and universal availability. FORTRAN afforded flexibility in all the areas considered. The analyst may choose the most efficient methods to reproduce stocastic phenomena. The methods used to represent the static and dynamic behavior of the system were limited only by the ingenuity and resourcefulness of the analyst. This flexibility, however, may be gained at a stiff price. The price paid for flexibility was paid in man hours spent in conceptualizing the problem, programming, monitoring and debugging the FORTRAN model. For the inexperienced analyst, flexibility may spell trouble. It is possible for the analyst to select a comparatively inefficient method, since many diversified methods are available to model a given system. FORTRAN, regardless of any disadvantages, will however still be used extensively for simulation. The reason being is its universal availability. FORTRAN compilers are available for nearly all commercially manufactured general purpose computers.

The principal advantage of GPSS is that the language is user oriented. The GPSS language assists the programmer in conceptualizing the problem, and the GPSS model required reduced programming, debugging and monitoring time compared to the FORTRAN model. The GPSS block diagram flow chart closely represented the simulated system, and the flow chart block mnemonics were quickly convertible to the GPSS

computer programming language. The above contributed significantly to reducing the time required to obtain a working simulation model. The primary disadvantage of GPSS is that output is restricted to bar graphs and frequency tables. It is conceivable that the GPSS output might have to be reformated by clerical personnel to conform to specific report requirements.

The problem formulation, programming and debugging time that was saved by GPSS, in contrast to FORTRAN, was of the order of three or four to one. This time differential is of such magnitude that the analyst should consider the use of GPSS even if the analyst has had no previous experience with the language. The advantages of GPSS over FORTRAN for simulation indicate that GPSS should be used whenever both GPSS and FORTRAN compilers are available.

It should be noted, however, that GPSS is basically restricted to the simulation of queueing systems while FORTRAN, as a general purpose language, can be used to simulate any system that can be described by a sequence of mathematical expressions.

APPENDIX A

Exogenous Data

The values utilized for exogenous data were hypothetical but conceptually realistic. One expects the mean length and arrival rate of messages to decrease as the precedence of the message increases. The actual mean values are normally a function of the type of military units within the geographical area serviced by the terminal. The arrival rate of messages is normally a function of the time of day. The arrival rate of messages generally peaks sometime after the end of the normal work day. Exogenous data that was utilized for both the FORTRAN and GPSS models is represented in Figures 5 thru 12 of this appendix.

PERCENT PEAK LOAD ARRIVAL RATE FOR TERMINAL ONE

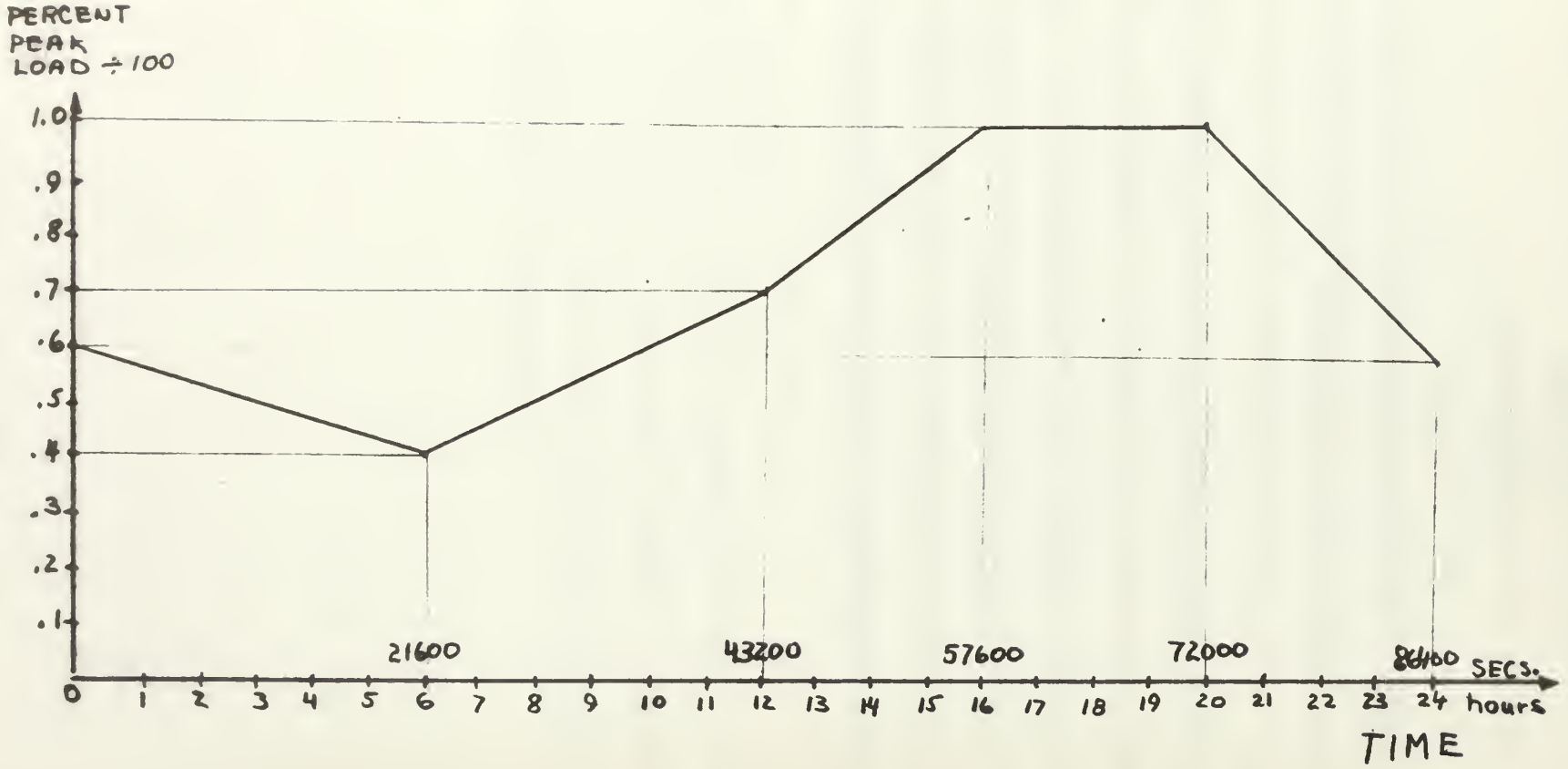


Figure 5

PERCENT PEAK LOAD ARRIVAL RATE FOR TERMINAL TWO

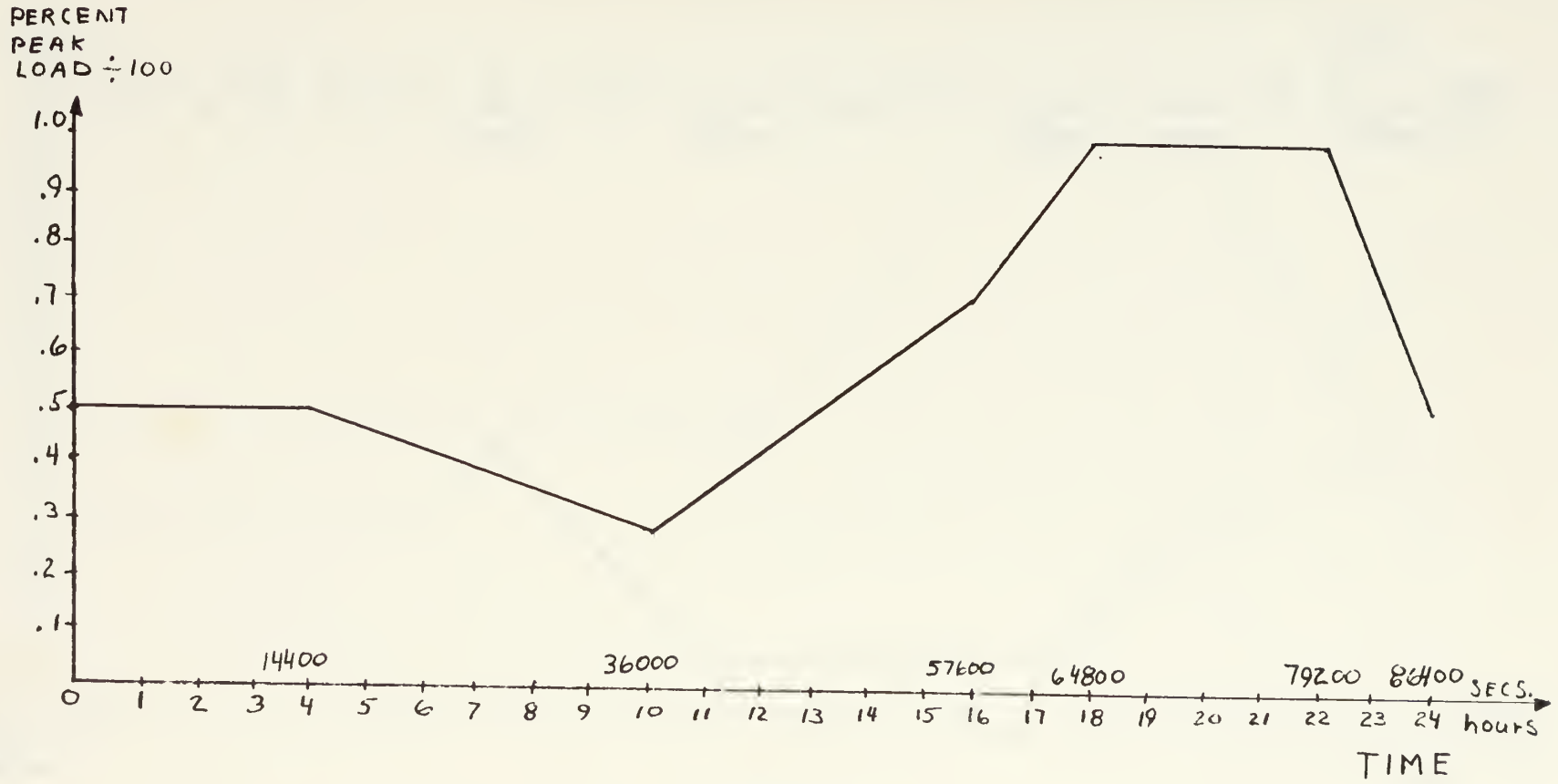


Figure 6

PERCENT PEAK LOAD ARRIVAL RATE FOR TERMINAL THREE

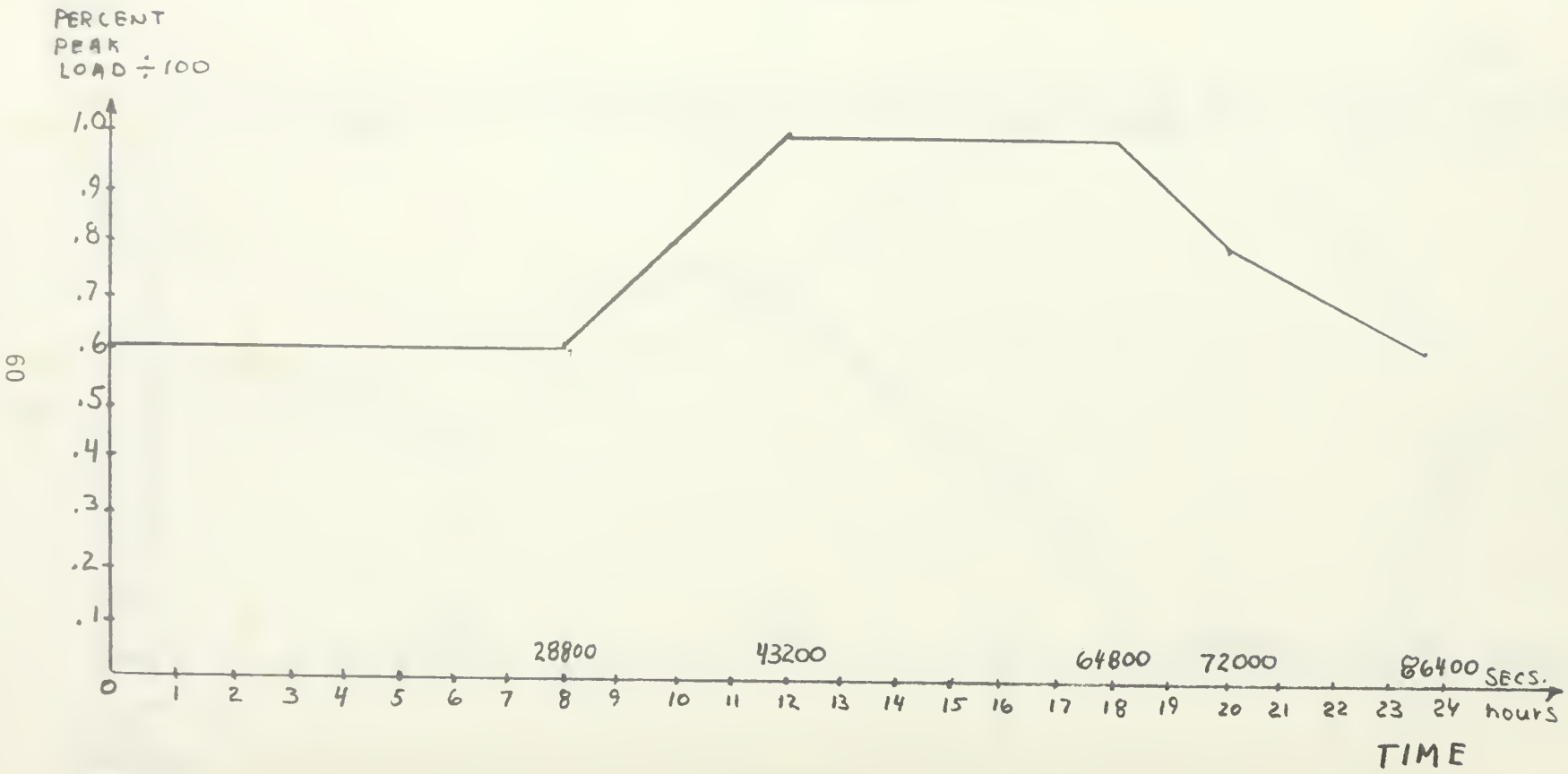


Figure 7

PERCENT PEAK LOAD ARRIVAL RATE FOR TERMINAL FOUR

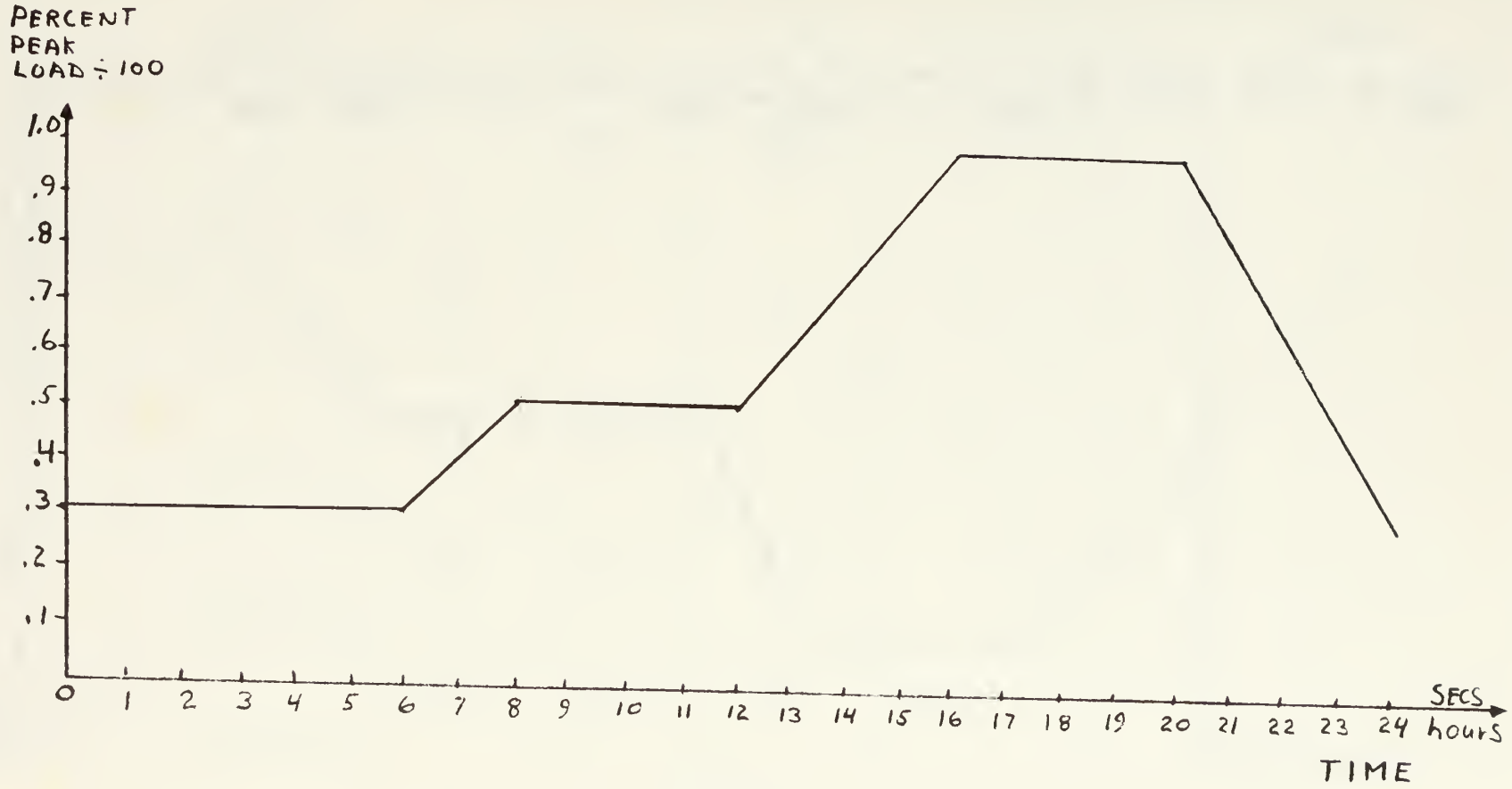


Figure 8

PERCENT PEAK LOAD ARRIVAL RATE FOR TERMINAL FIVE

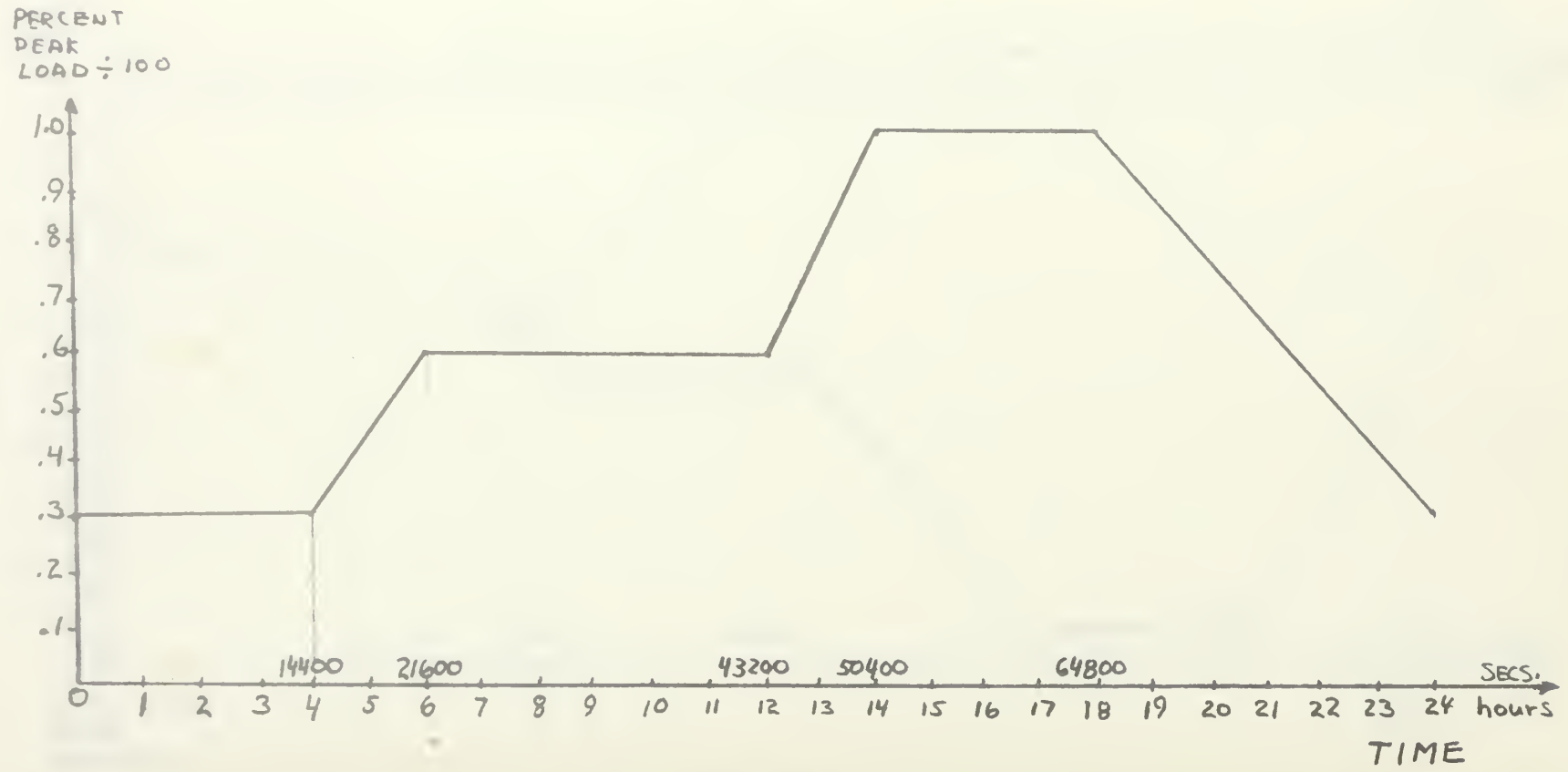


Figure 9

PERCENT TRAFFIC TRANSMITTED TO EACH TERMINAL

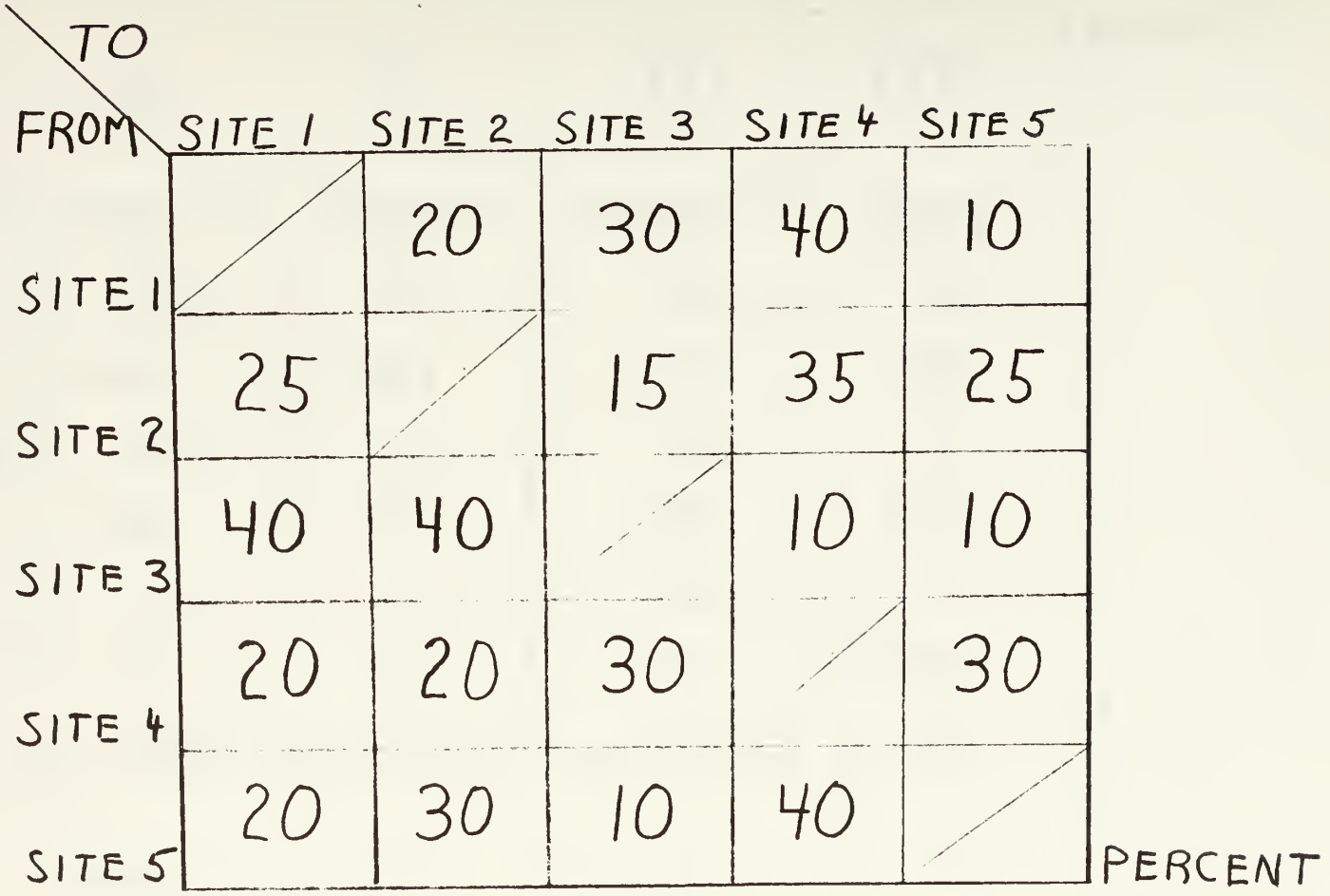


Figure 10

MEAN PEAK HOUR ARRIVAL RATE BY PRECEDENCE

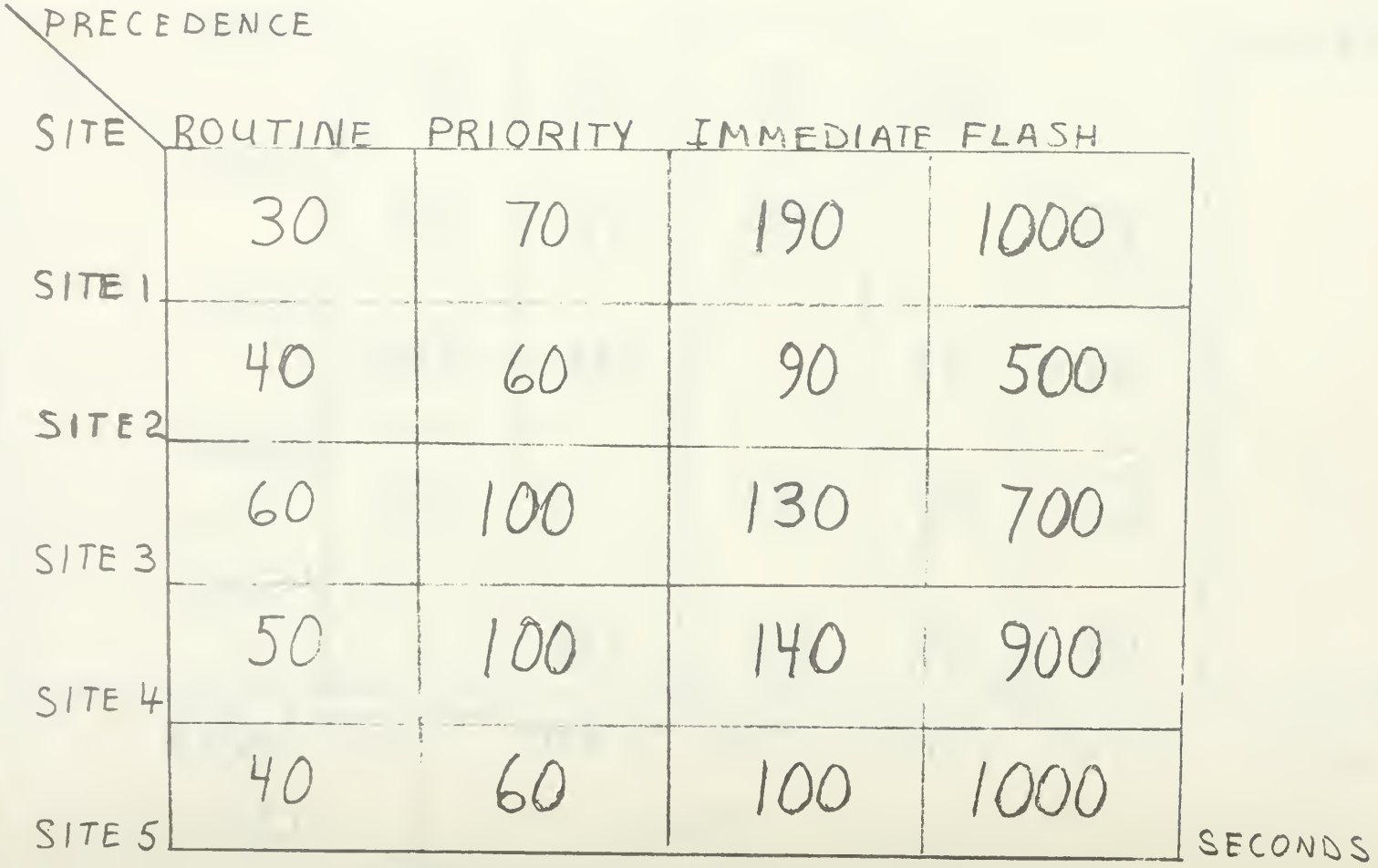


Figure 11

MEAN MESSAGE TRANSMISSION TIME BY PRECEDENCE

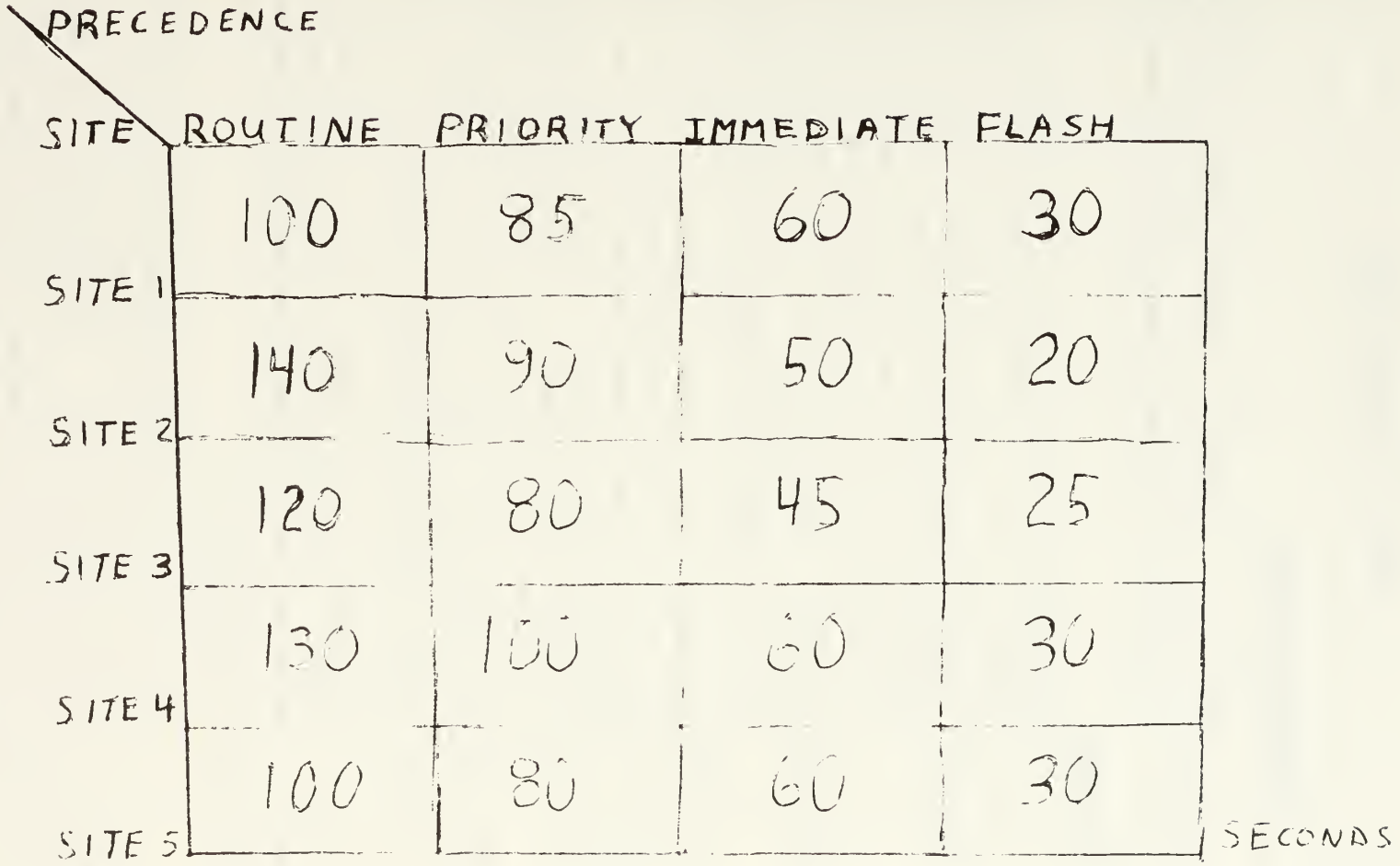


Figure 12

APPENDIX B

FORTRAN Computer Model

A. Definition of Arrays

1. ARTE(4,5) contained the mean peak hour arrival rates listed in Figure 11,Appendix A. Columns corresponded to terminals 1 thru 5, respectively.

2. BACKLP(10) stored the maximum low precedence backlog of messages in minutes for each terminal and relay transmitter bay.

3. BACKHP(10) stored the maximum high precedence backlog of messages in minutes for each terminal and relay transmitter bay.

4. FF(5,10,3) stored flash message parameters for the terminals and relay transmit bays. Rows 1 thru 5 stored arrival times, and position FF(1,1,1) contained the lowest arrival time for terminal one. Columns 1 thru 10 represented terminal stations 1 thru 5, and relay transmitter bays 6 thru 10, respectively. Relay transmitter bay 6 transmitted messages to terminal 1, bay 7 transmitted messages to terminal 2, etc. The third dimension in the array was used to store message attributes as follows: Position 1 was arrival time, position 2 was message transmission time and position 3 was destination.

5. HOLD1(i) - HOLD10(i) represented the circuits between terminals and relay transmitter bays. The mnmonics of the HOLD arrays corresponded to the numbering depicted in Figure 1, Section B, Chapter III. The dimension of each array was equal to the number of transmit circuits between two stations and was defined by variables H1 thru H10.

6. NEXEV(20,2) was the array used to store the next event times for the terminals, relay transmitter bays and groups of transmitter circuits. The transmitter circuits were represented by arrays HOLD1 thru HOLD10. A message that was transmitted was stored in a HOLD array for the transmission length of the message. Column 1 of array NEXEV recorded the next event time and column 2 was a statement number corresponding to the subroutine call statement associated with that event.

7. OO(20,10,3) stored immediate message parameters for the terminals and relay transmit bays. Rows 1 thru 20 stored arrival times and position OO(1,1,1) contained the lowest arrival time for terminal one. Columns 1 thru 10 represented terminal stations 1 thru 5, and relay transmitter bays 6 thru 10, respectively. Relay transmitter bay 6 transmitted messages to terminal 1, bay 7 transmitted messages to terminal 2, etc. The third dimension in the array was used to store message attributes as follows. Position 1 was arrival time, position 2 was message transmission time, and position 3 was destination.

8. PP(30,10,3) stored priority messages for the terminals and relays. Rows 1 thru 30 stored arrival times and position PP(1,1,1) contained the lowest arrival time for terminal one. Columns 1 thru 10 represented terminal stations 1 thru 5, and relay transmitter bays 6 thru 10, respectively. Relay transmitter bay 6 transmitted messages to terminal 1, bay 7 transmitted messages to terminal 2, etc. The third dimension in the array was used to store message attributes as follows: Position 1 was arrival time, position 2 was message transmission time, and position 3 was destination.

9. RR(60,10,3) stored routine messages for the terminals and relays. Rows 1 thru 60 stored arrival times and position RR(1,1,1) contained the lowest arrival time for terminal one. Columns 1 thru 10 represented terminal stations 1 thru 5, and relay transmitter bays 6 thru 10, respectively. Relay transmitter bay 6 transmitted messages to terminal 1, bay 7 transmitted messages to terminal 2, etc. The third dimension in the array was used to store message attributes as follows: Position 1 was arrival time, position 2 was message transmission time, and position 3 was destination.

10. ZLY(8,5) was initialized to contain the percent peak load for each station as represented in Figures 5 thru 9, Appendix A.

11. ZLX(8,5) was initialized to contain the time of day corresponding to percent peak load for stations 1 thru 5 as represented in Figures 5 thru 9, Appendix A.

12. ZPTT(5,5) was initialized to contain the percent of messages transmitted among stations, as depicted by Figure 10, Appendix A.

B. Subroutines

1. BACKLOG(K)

a. Arguments

K was a terminal site number or relay transmitter bay number, as depicted by Figure 1, Section B, Chapter III.

b. Purpose

Subroutine BACKLOG maintained statistics on the maximum high and low precedence backlog for each terminal and for each relay transmitter bay.

2. DEST(J,DES)

a. Arguments

J was a value 1,2,3,4 or 5 that represented a terminal designation. DES was a number 1 thru 5 corresponding to the destination terminal of the message.

b. Purpose

Subroutine DEST obtained a pseudo-random number and in conjunction with array ZPTT determined the destination of an originated message.

3. FILL(K)

a. Argument

Argument K corresponded to the terminal designators 1,2,3,4 and 5.

b. Purpose

Subroutine FILL created a specified number of messages of each precedence to initially fill arrays RR, PP, OO and FF. Message transmission time and destination were also generated for each message.

4. IHOLD(HOLD,L,M)

a. Arguments

HOLD was an array of dimension determined by variables H1 thru H10. The dimension equaled the number of circuits between two stations. L was the dimension of array HOLD in the subroutine. M was the row number in array NEXEV(20,2) corresponding to the event under consideration.

b. Purpose

Subroutine IHOLD was used to release a circuit to availability after a message was completely transmitted. If two events had equal next event times, subroutine IHOLD was called prior to subroutines TERM or RELAY.

5. INTER(K,TIME,ZNS)

a. Arguments

K was a value 1 thru 5 corresponding to a terminal designator. TIME was the time of day in seconds. ZNS was the percent of peak load corresponding to the argument TIME.

b. Purpose

Subroutine INTER performed a linear interpolation on Figures 1 thru 5 of Appendix A. INTER produced a percent peak load value for a given time of day.

6. RANDO(IY,YFL)

a. Arguments

IX was in FORTRAN common with the main program and was initialized an odd integer number. After the initial value had been specified, IX became a function of IY which also was an integer. YFL was a pseudo-random number in the range (0,1).

b. Purpose

Subroutine RANDO generated pseudo-random numbers in the range (0,1).

7. RELAY(HOLD,K,L,M)

a. Arguments

HOLD was an array of dimension L. L was equal to the number of transmit circuits. K was the row number in array NEXEV (20,2) corresponding to the event under consideration.

b. Purpose

RELAY processed messages being transmitted from a relay transmitter bay to a terminal. The subroutine considered messages of precedence flash, immediate, priority, and routine on a first in first out discipline by precedence. Flash messages pre-empted a lower precedence message if no circuit was available. The message selected was transferred to the respective HOLD array where it remained for its transmission time. The backlogged messages at the terminal then advanced one position in their respective array columns. Necessary statistics were compiled to determine backlogs and circuit utilization. The next event time for the given terminal was set in accordance with circuit availability.

8. RFILL(K,MOVE,PREC)

a. Arguments

K corresponded to the terminal station numbers 1,2,3,4,5 and relay transmitted bay numbers 6,7,8,9,10. If MOVE equaled 1, RFILL was called by a relay and if 2, RFILL was called by a terminal.

b. Purpose

Subroutine RFILL had two functions. For both the relay and terminal, the subroutine advanced quequed messages one position in the message's respective precedence array column. After the messages had been advanced, at least the last position in the array was vacent. For a terminal, a new message was generated and was placed in the last position of the precedence array column.

9. RELREO(K,PREC)

a. Arguments

K referred to relay transmit bay positions, and assumed values 6,7,8,9 and 10. PREC assumed values of 1,2,3 and 4 that corresponded to precedences routine, priority, immediate and flash, respectively.

b. Purpose

Subroutine RELREO ordered message arrival times in relay transmit bay columns of the precedence arrays. This was done in order to assure that a message with the lowest arrival time was in position one for each transmit bay.

10. TERM(HOLD,K,L,M)

a. Arguments

HOLD was an array of dimension L. L was the number of transmit circuits. K was the row number in array NEXEV (20,2) corresponding to the event under consideration.

b. Purpose

Subroutine TERM processed messages that were transmitted from a terminal to a relay. The subroutine considered messages of precedence flash, immediate, priority and routine on a first in first out discipline by precedence. Flash messages pre-empt a lower precedence message if no free circuit was available. A message was transferred to its respective HOLD array and to a future events list in queue in the relay transmitter bay that served the desired destination. The backlogged messages at the terminal advanced one position, and then a new message was added to the future event list. Necessary statistics were compiled to determine backlogs and circuit utilization. The next event time for the given terminal was set in accordance with circuit availability.

11. UTIL(CLOCK)

a. Arguments

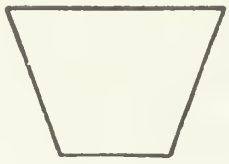
Clock was a constant equal to the length of computer run in simulation time units, divided by 1,000.

b. Purpose

UTIL computed the average circuit utilization for each terminal and relay bay to three figures.

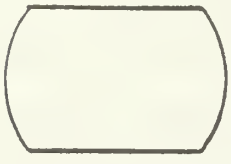
FORTRAN FLOW CHART SYMBOLS

ENTRY OR EXIT POINT

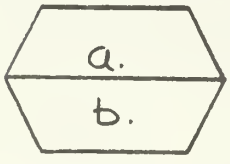


INPUT / OUTPUT OPERATION

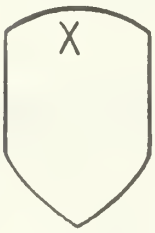
CALCULATION OR INSTRUCTION



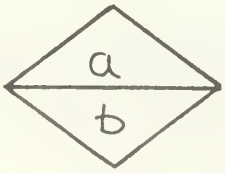
DECISION POINT



a. SUBROUTINE NAME
b. NARRATIVE DESCRIPTION PAGE



COMPUTED GO TO ON VARIABLE X

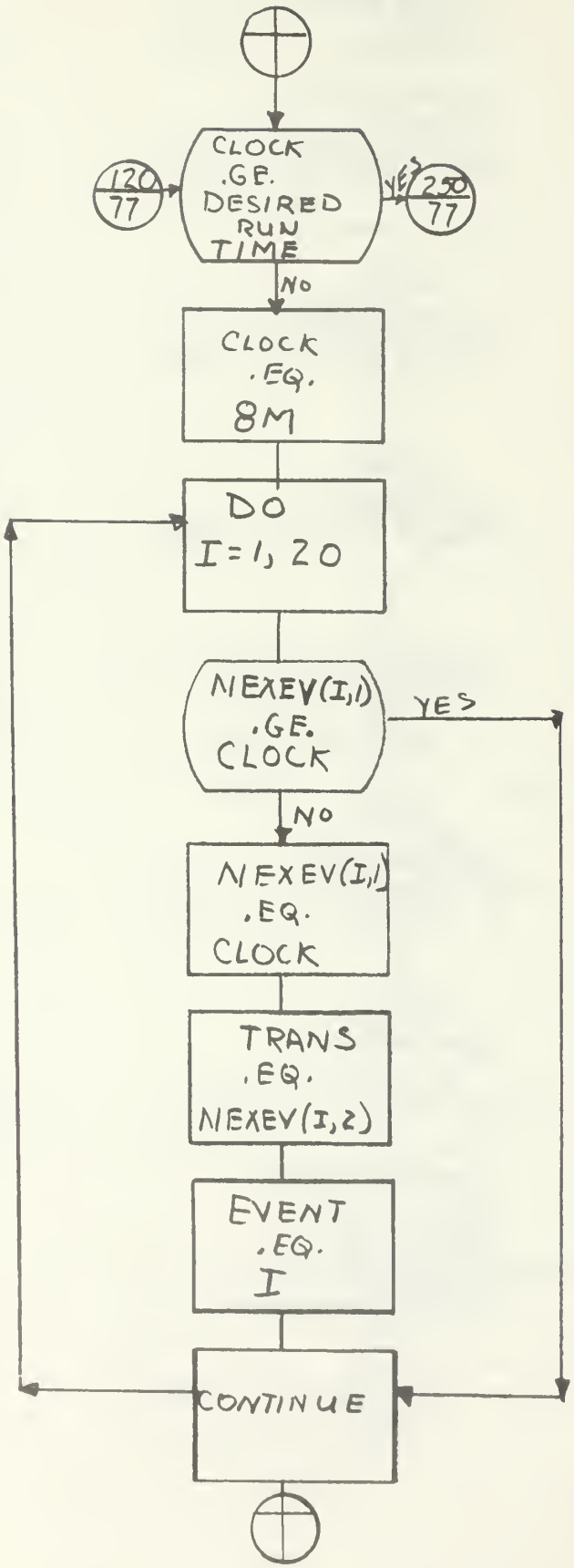


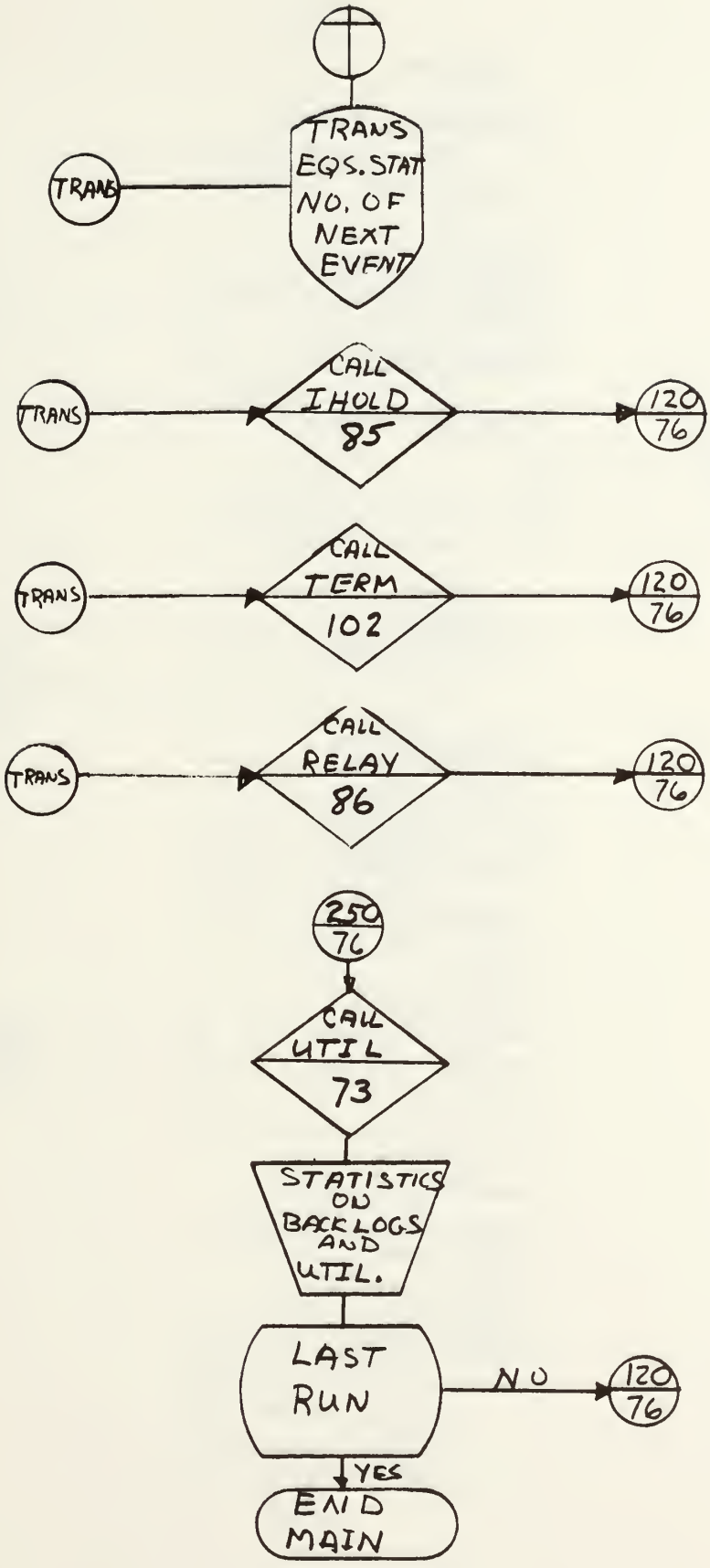
a. CALL STATEMENT
b. PAGE NUMBER

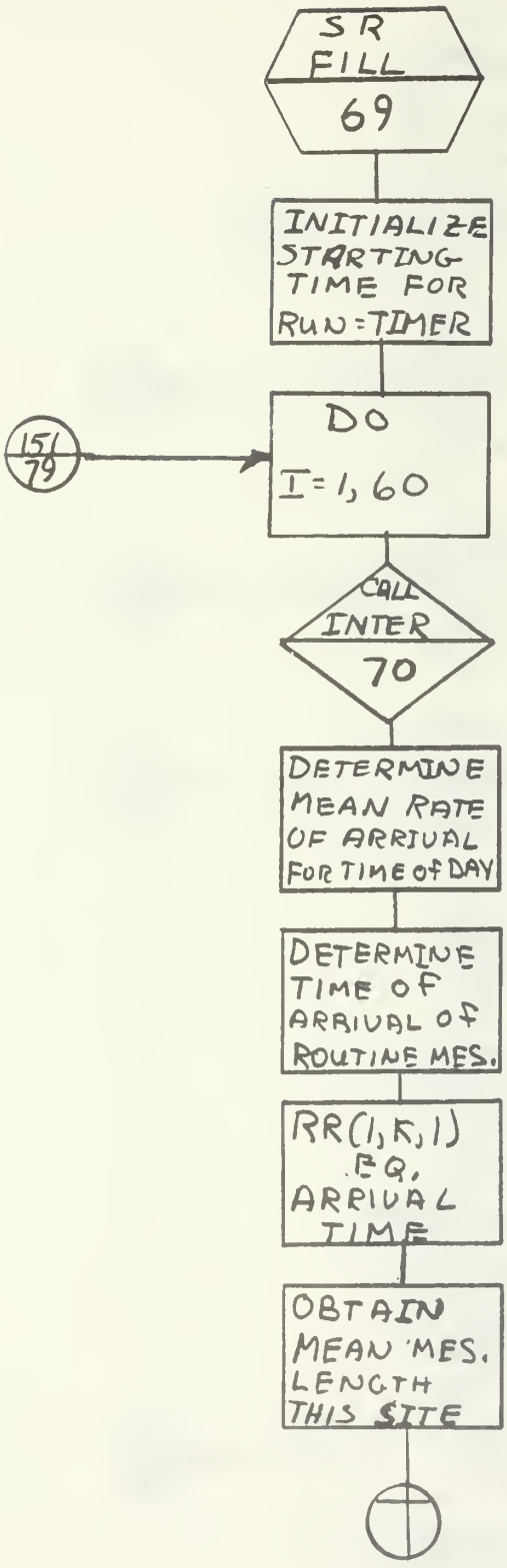
a. TRANSFER NUMBER
b. PAGE NUMBER

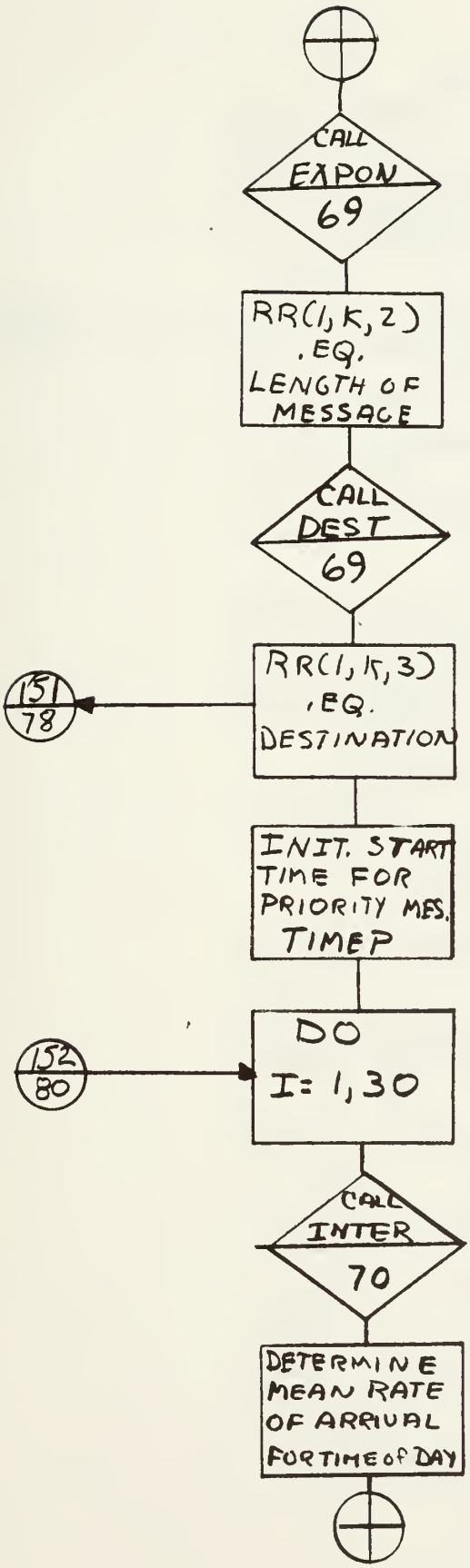
OFF PAGE CONNECTOR

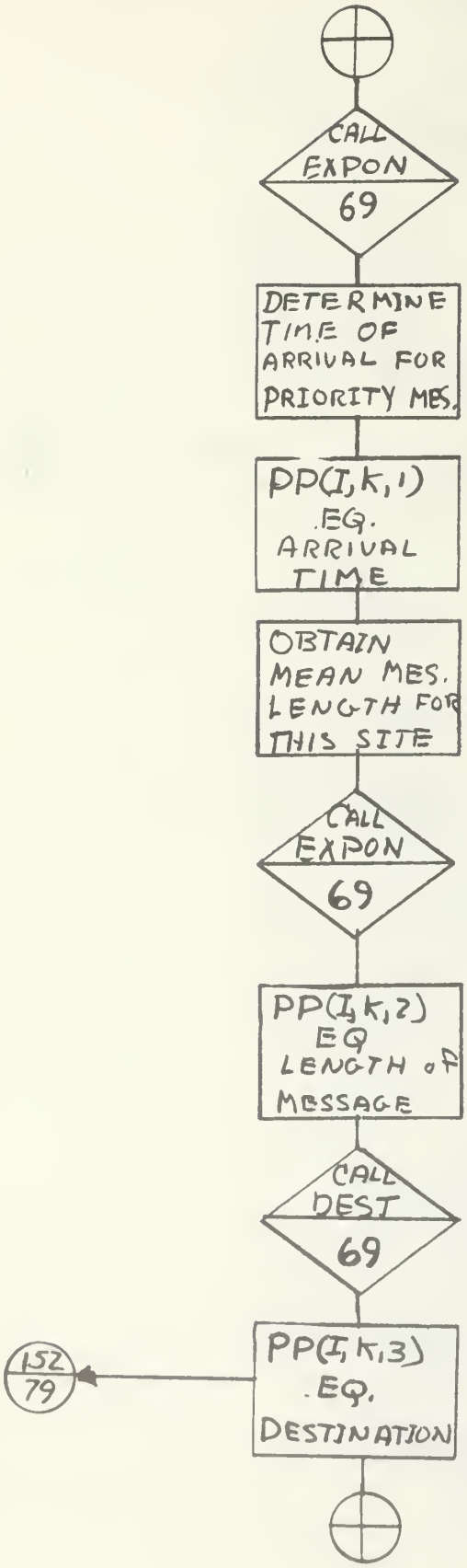
FORTRAN FLOW CHART

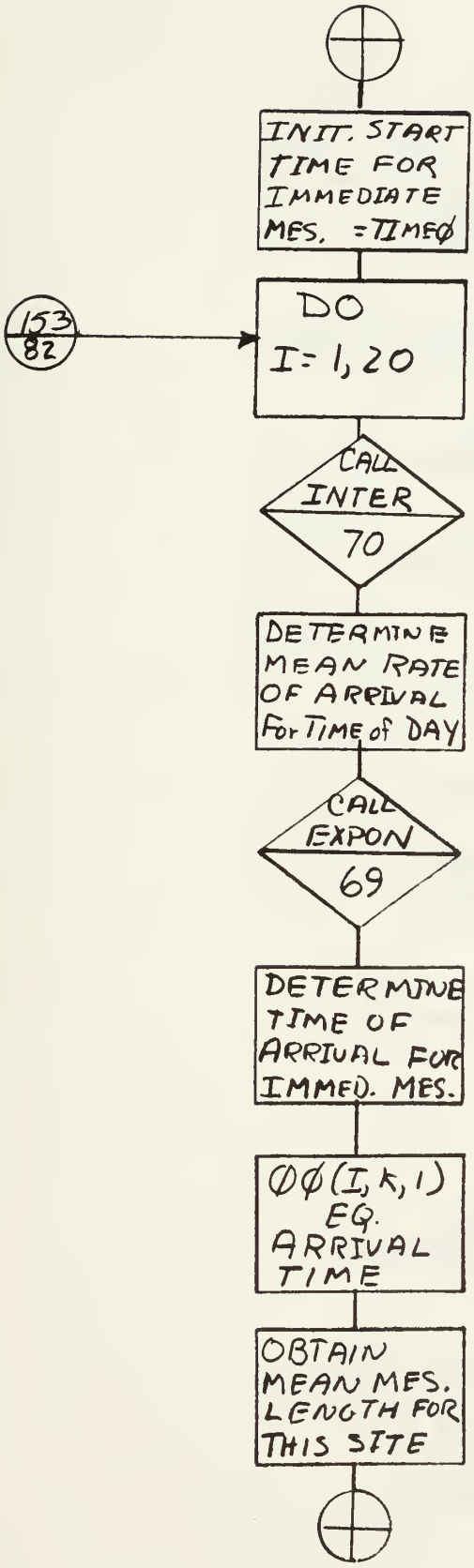

CLOCK
.GE.
DESIRED
RUN
TIME
120
77
YES
250
77
No
CLOCK
.EQ.
8M
DO
I=1, 20
NEXEV(I,1)
.GE.
CLOCK
YES
No
NEXEV(I,1)
.EQ.
CLOCK
TRANS
.EQ.
NEXEV(I,2)
EVENT
.EQ.
I
CONTINUE

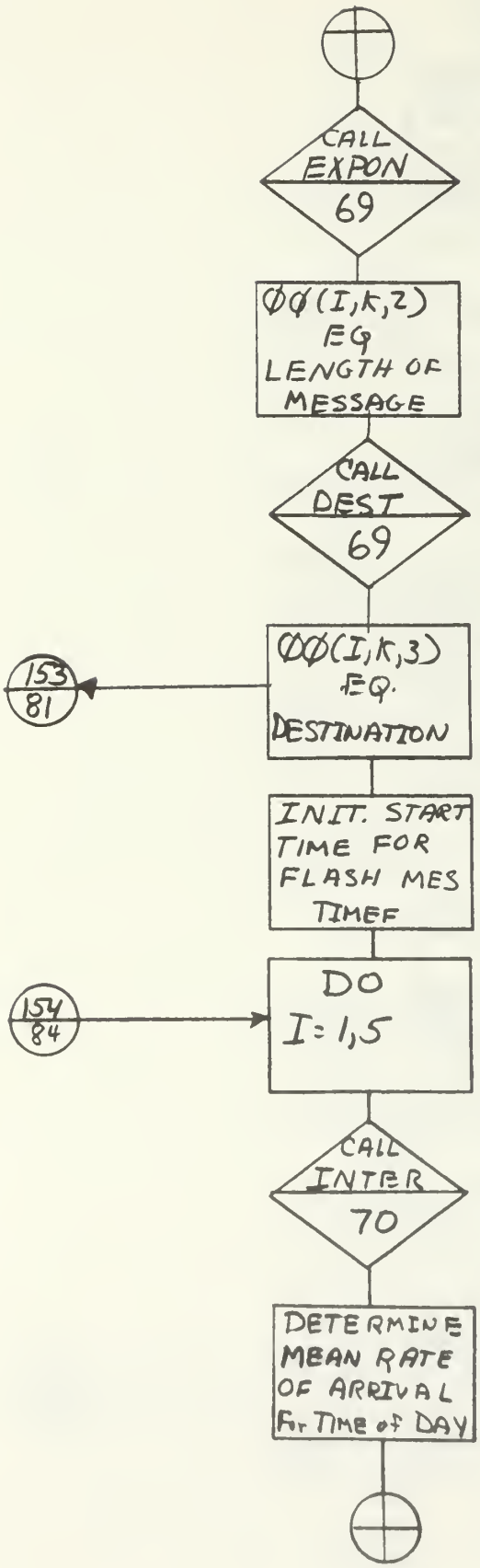

TRANS
TRANS EQS. STAT NO. OF NEXT EVENT
TRANS
CALL IHOLD 85
120 76
TRANS
CALL TERM 102
120 76
TRANS
CALL RELAY 86
120 76
250 76
CALL UTIL 73
STATISTICS ON BACKLOGS AND UTIL.
LAST RUN
NO
120 76
YES
END MAIN

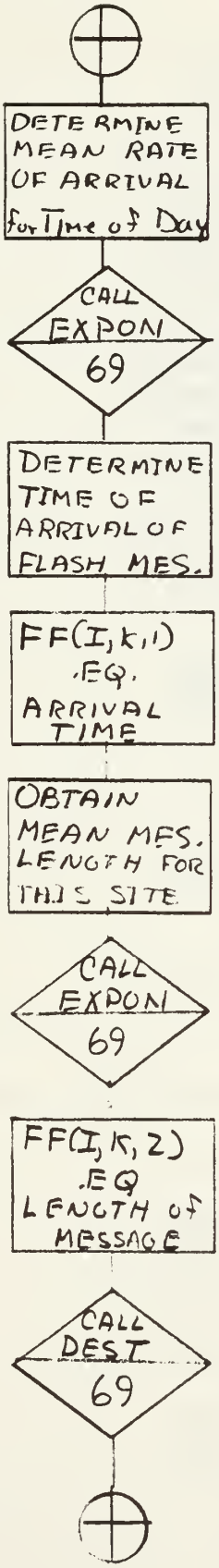

SR
FILL
69
INITIALIZE STARTING TIME FOR RUN=TIMER
DO
I=1,60
157
79
CALL
INTER
70
DETERMINE MEAN RATE OF ARRIVAL FOR TIME of DAY
DETERMINE TIME OF ARRIVAL of ROUTINE MES.
RR(1,K,1) EQ. ARRIVAL TIME
OBTAIN MEAN MES. LENGTH THIS SITE

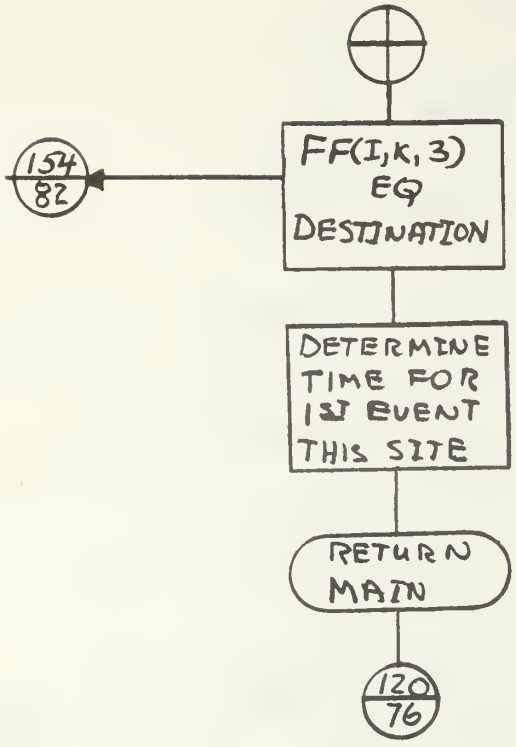

CALL EXPON 69
RR(1,K,2) .EQ. LENGTH OF MESSAGE
CALL DEST 69
151 78
RR(1,K,3) .EQ. DESTINATION
INIT. START TIME FOR PRIORITY MES. TIMEP
152 80
DO I= 1,30
CALL INTER 70
DETERMINE MEAN RATE OF ARRIVAL FOR TIME OF DAY

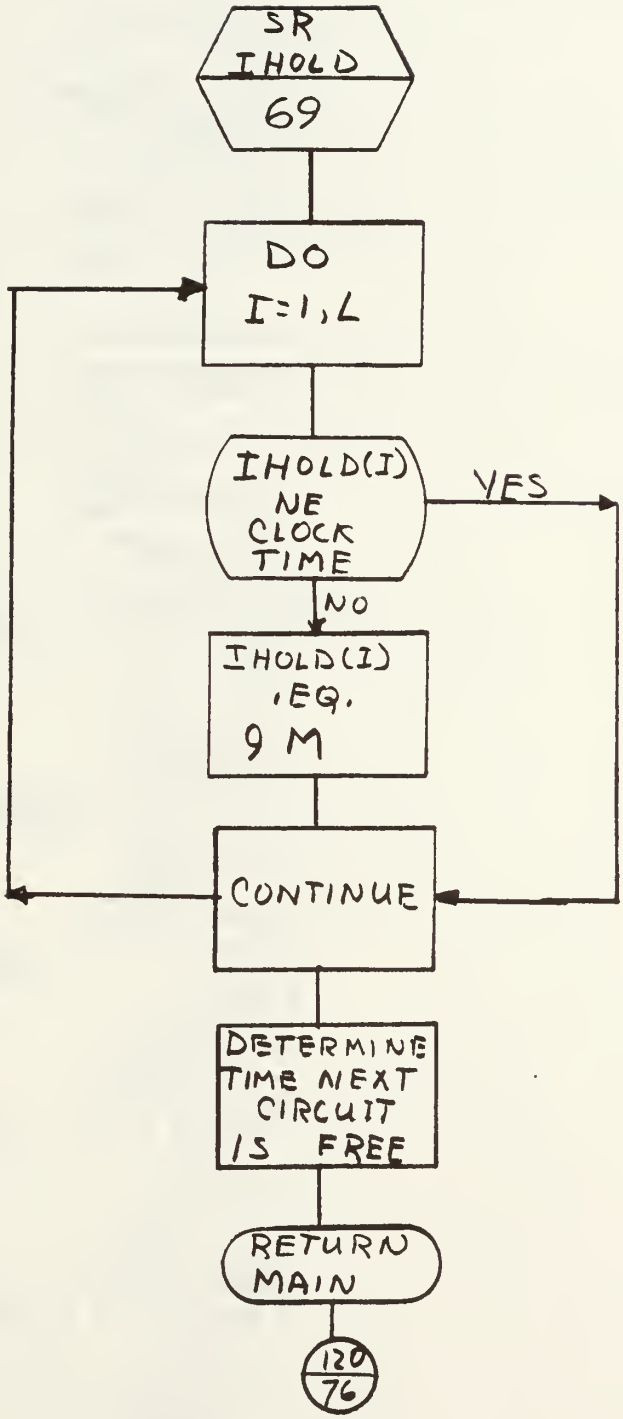

CALL EXPON 69
DETERMINE TIME OF ARRIVAL FOR PRIORITY MES.
PP(I,K,1) EQ. ARRIVAL TIME
OBTAIN MEAN MES. LENGTH FOR THIS SITE
CALL EXPON 69
PP(I,K,2) EQ LENGTH OF MESSAGE
CALL DEST 69
PP(I,K,3) EQ. DESTINATION
152/79

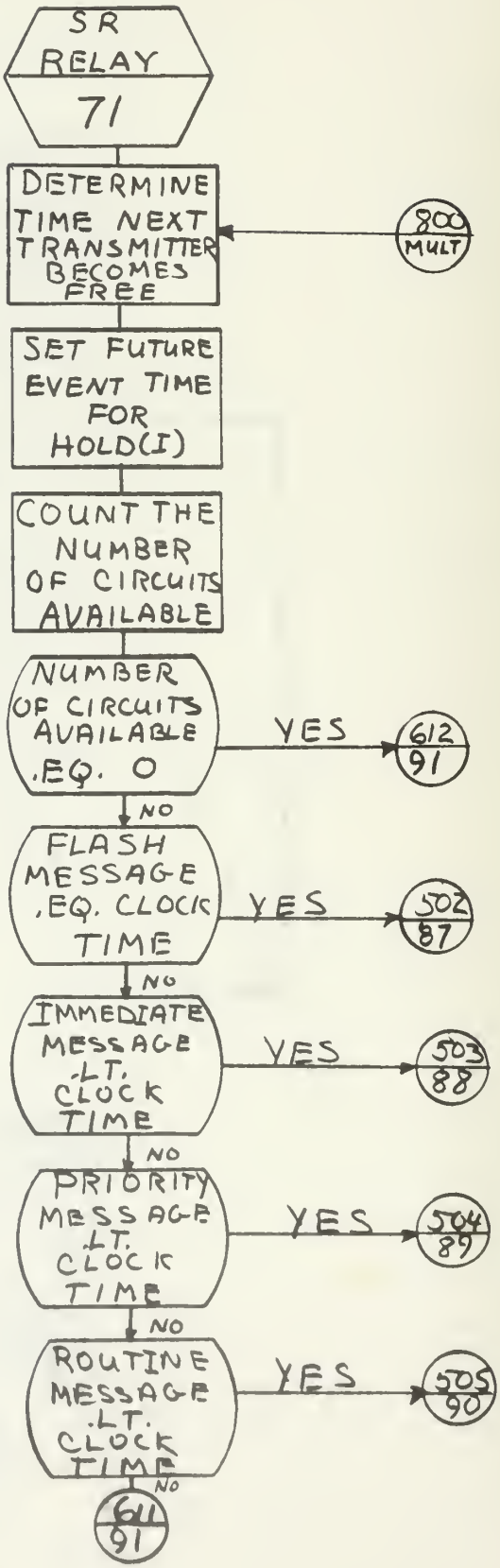

INIT. START TIME FOR IMMEDIATE MES. =TIMEØ
153/82
DO I=1,20
CALL INTER 70
DETERMINE MEAN RATE OF ARRIVAL FOR TIME of DAY
CALL EXPON 69
DETERMINE TIME OF ARRIVAL FOR IMMED. MES.
ØØ(I,K,1) EQ. ARRIVAL TIME
OBTAIN MEAN MES. LENGTH FOR THIS SITE


CALL
EXPON
69
ØØ(I,K,2)
EQ
LENGTH OF
MESSAGE
CALL
DEST
69
153
81
ØØ(I,K,3)
EQ.
DESTINATION
INIT. START
TIME FOR
FLASH MES
TIMEF
154
84
DO
I=1,5
CALL
INTER
70
DETERMINE
MEAN RATE
OF ARRIVAL
Fr TIME of DAY

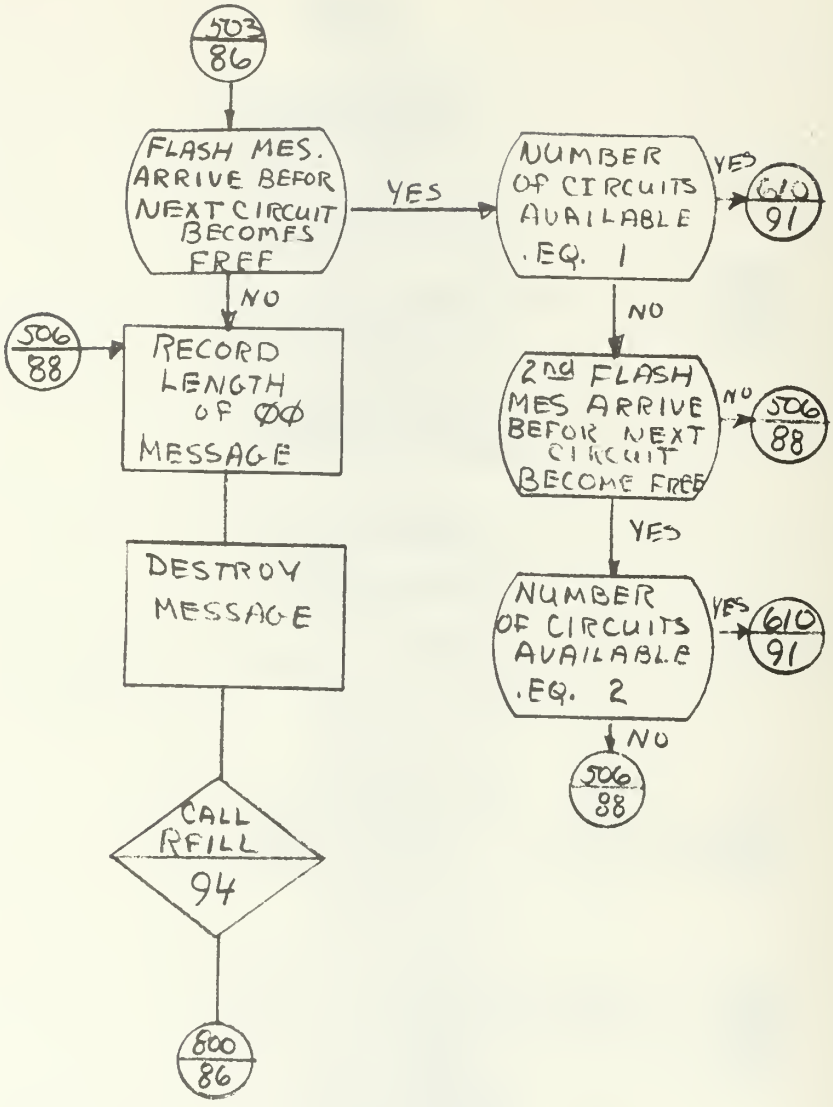

DETERMINE MEAN RATE OF ARRIVAL for Time of Day
CALL EXPON 69
DETERMINE TIME OF ARRIVAL OF FLASH MES.
FF(I,K,1) .EQ. ARRIVAL TIME
OBTAIN MEAN MES. LENGTH FOR THIS SITE
CALL EXPON 69
FF(I,K,2) .EQ LENGTH of MESSAGE
CALL DEST 69

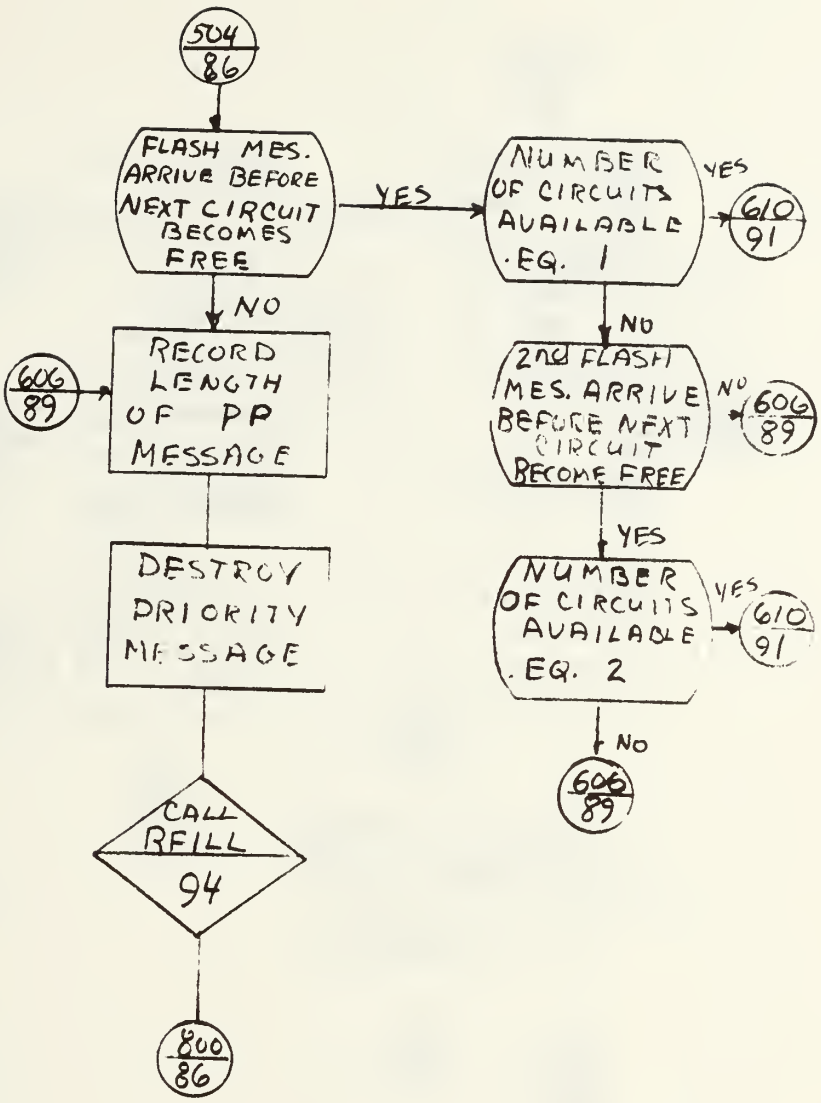

154
82
FF(I,K,3)
EQ
DESTINATION
DETERMINE
TIME FOR
IJ EVENT
THIS SITE
RETURN
MAIN
120
76

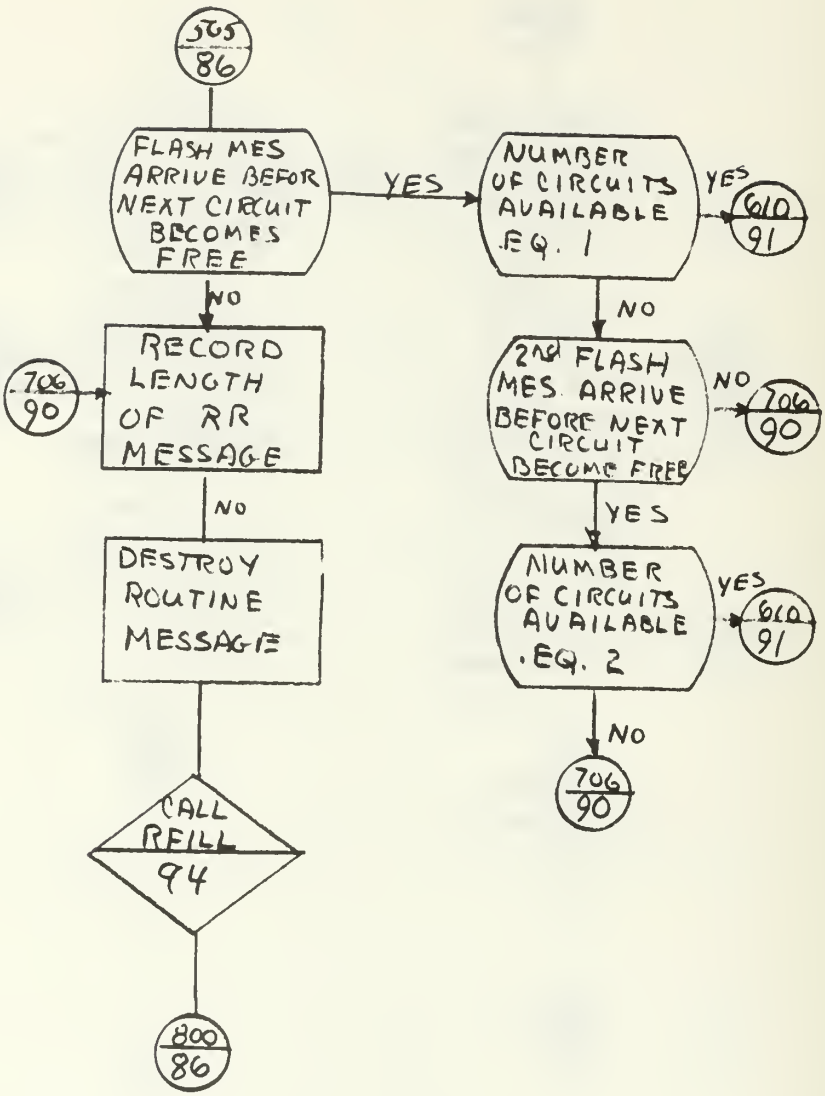

SR
IHOLD
69
DO
I=1,L
IHOLD(I)
NE
CLOCK
TIME
YES
NO
IHOLD(I)
.EQ.
9 M
CONTINUE
DETERMINE
TIME NEXT
CIRCUIT
IS FREE
RETURN
MAIN
120
76

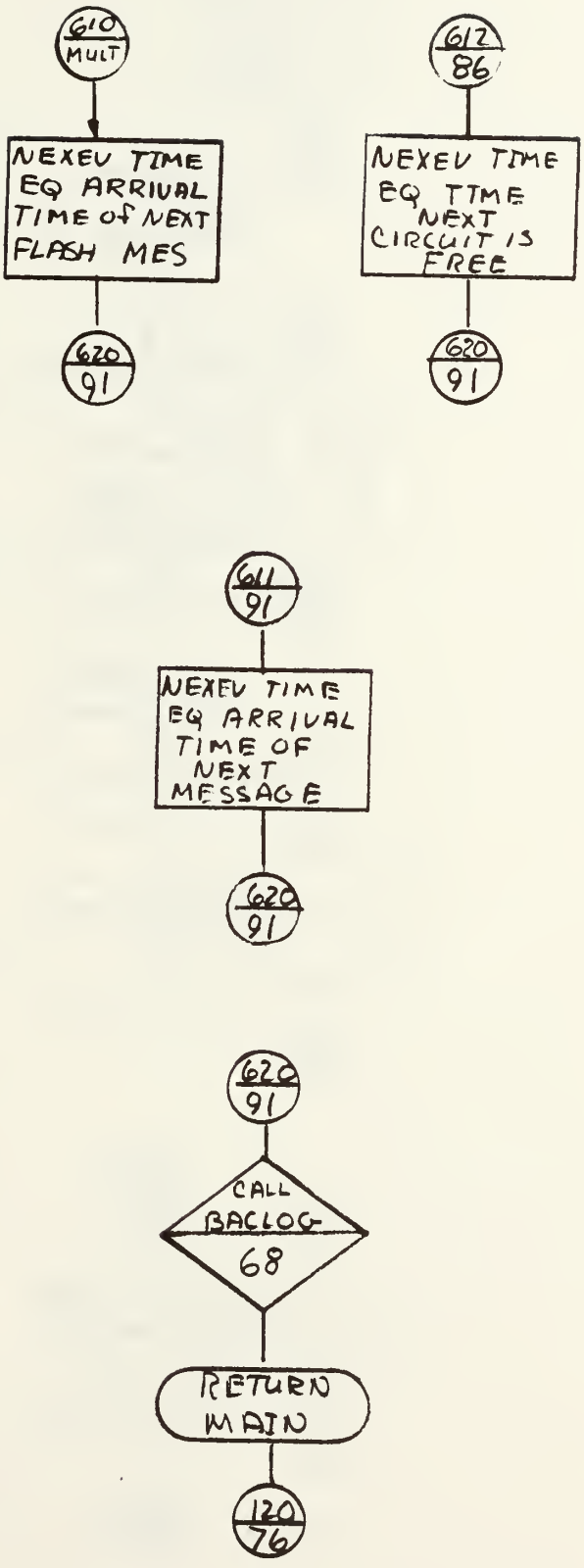

SR
RELAY
71
DETERMINE TIME NEXT TRANSMITTER BECOMES FREE
800
MULT
SET FUTURE EVENT TIME FOR HOLD(I)
COUNT THE NUMBER OF CIRCUITS AVAILABLE
NUMBER OF CIRCUITS AVAILABLE .EQ. 0
YES
612
91
NO
FLASH MESSAGE .EQ. CLOCK TIME
YES
502
87
NO
IMMEDIATE MESSAGE .LT. CLOCK TIME
YES
503
88
NO
PRIORITY MESSAGE .LT. CLOCK TIME
YES
504
89
NO
ROUTINE MESSAGE .LT. CLOCK TIME
YES
505
90
NO
611
91

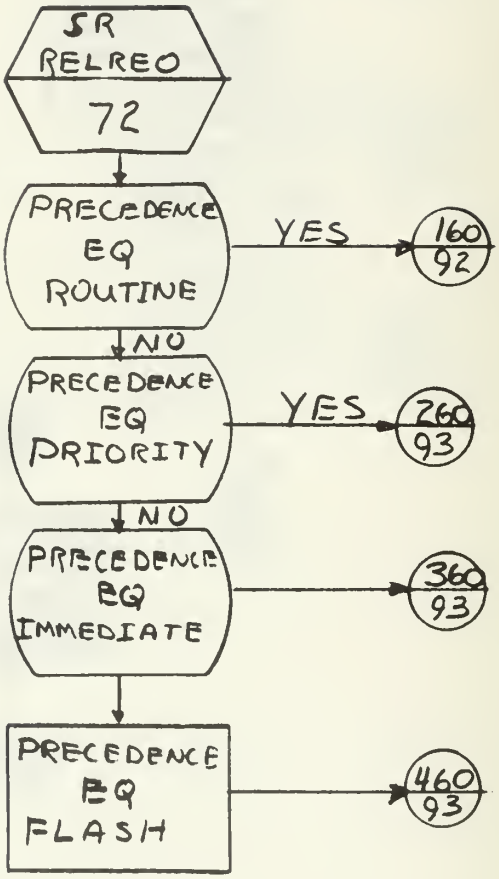

502
86
RESERVE A XMTR FOR MESSAGE LENGTH
RECORD LENGTH OF MESSAGE
DESTROY MESSAGE
CALL RFILL
94
RFILL MOVES A MESSAGE UP TO FF(I,K,1)
800
86

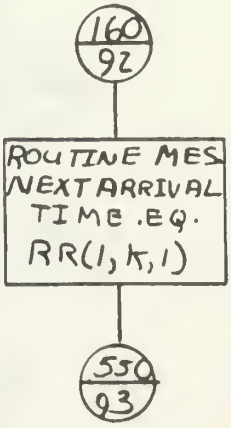

503
86
FLASH MES. ARRIVE BEFOR NEXT CIRCUIT BECOMES FREE
YES
NUMBER OF CIRCUITS AVAILABLE .EQ. 1
YES
610
91
NO
NO
506
88
RECORD LENGTH OF ØØ MESSAGE
2nd FLASH MES ARRIVE BEFOR NEXT CIRCUIT BECOME FREE
NO
506
88
YES
DESTROY MESSAGE
NUMBER OF CIRCUITS AVAILABLE .EQ. 2
YES
610
91
NO
506
88
CALL RFILL
94
800
86

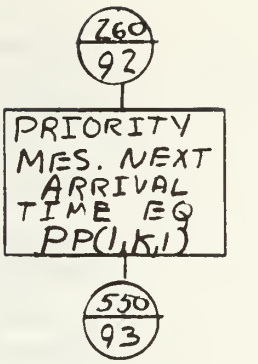

504/86
FLASH MES. ARRIVE BEFORE NEXT CIRCUIT BECOMES FREE
YES
NUMBER OF CIRCUITS AVAILABLE .EQ. 1
YES
610/91
NO
NO
606/89
RECORD LENGTH OF PP MESSAGE
2nd FLASH MES. ARRIVE BEFORE NEXT CIRCUIT BECOME FREE
NO
606/89
YES
DESTROY PRIORITY MESSAGE
NUMBER OF CIRCUITS AVAILABLE .EQ. 2
YES
610/91
No
606/89
CALL REILL 94
800/86

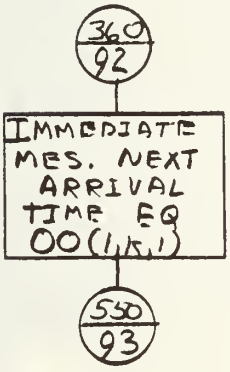

505
86
FLASH MES ARRIVE BEFOR NEXT CIRCUIT BECOMES FREE
YES
NUMBER OF CIRCUITS AVAILABLE EQ. 1
YES
610
91
NO
NO
RECORD LENGTH OF RR MESSAGE
706
90
2ND FLASH MES ARRIVE BEFORE NEXT CIRCUIT BECOME FREE
NO
706
90
YES
NO
DESTROY ROUTINE MESSAGE
NUMBER OF CIRCUITS AVAILABLE EQ. 2
YES
610
91
NO
706
90
CALL REFILL
94
800
86

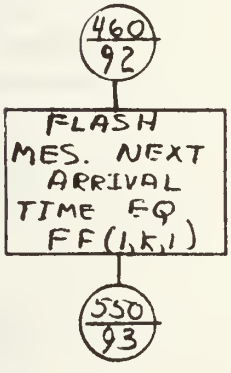

610
MULT
NEXEV TIME EQ ARRIVAL TIME of NEXT FLASH MES
620
91
612
86
NEXEV TIME EQ TTME NEXT CIRCUIT IS FREE
620
91
611
91
NEXEV TIME EQ ARRIVAL TIME OF NEXT MESSAGE
620
91
620
91
CALL BACLOG
68
RETURN MAIN
120
76

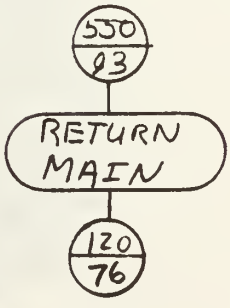

SR
RELREQ
72
PRECEDENCE
EQ
ROUTINE
YES
160
92
NO
PRECEDENCE
EQ
PRIORITY
YES
260
93
NO
PRECEDENCE
EQ
IMMEDIATE
360
93
PRECEDENCE
EQ
FLASH
460
93

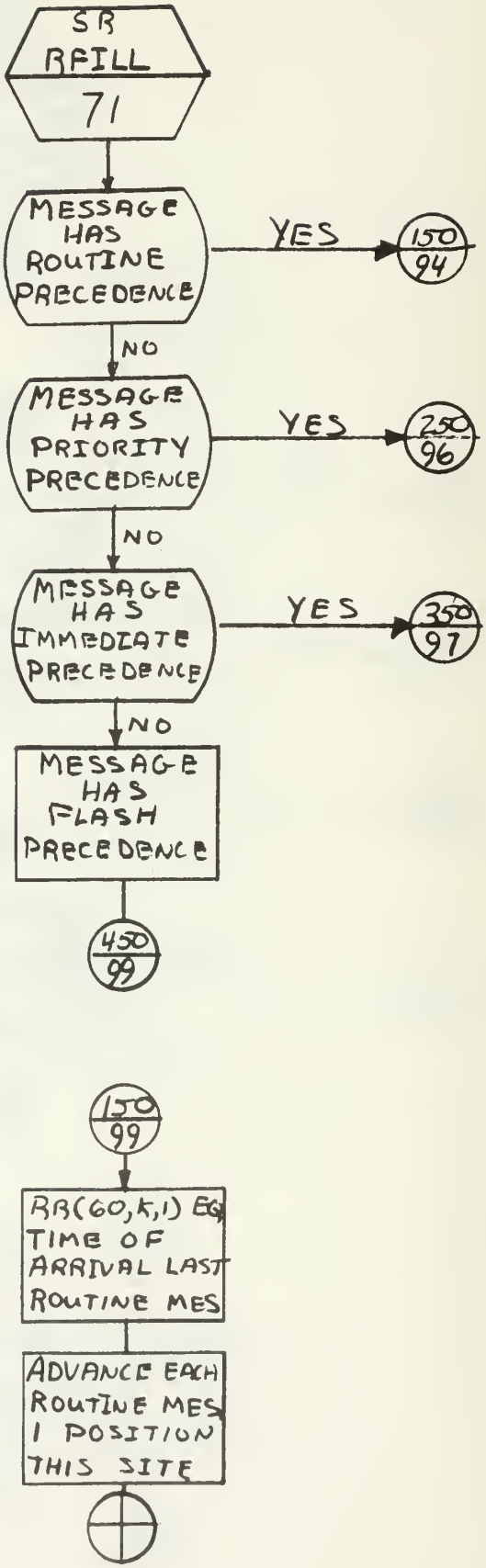

160
92
ROUTINE MES
NEXT ARRIVAL
TIME .EQ.
RR(I,K,1)
550
93

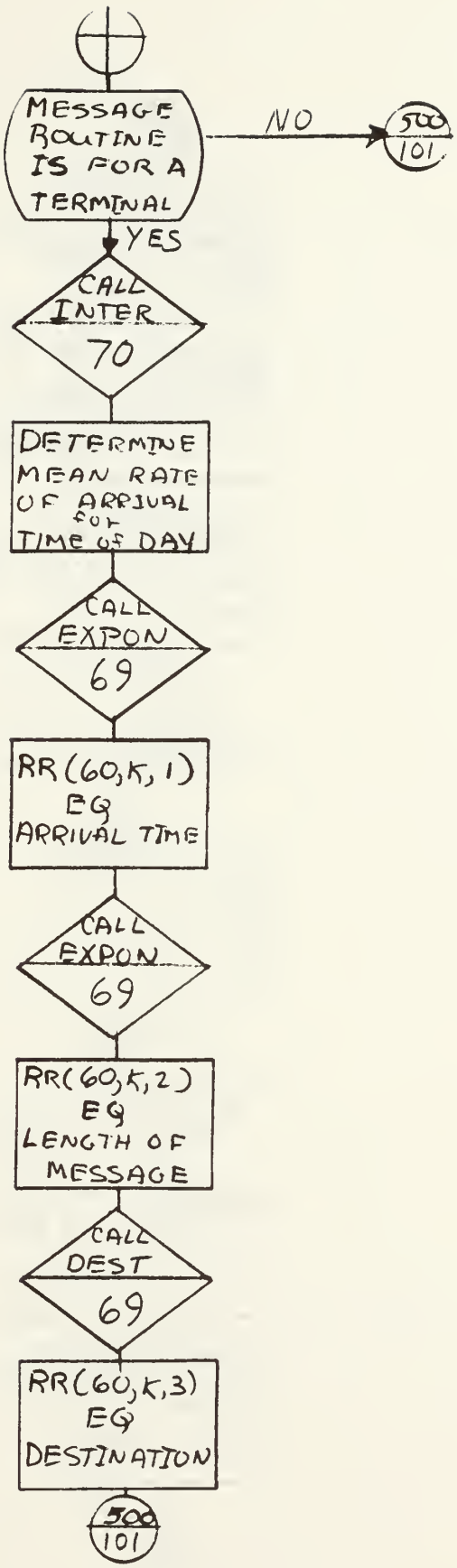

260
92
PRIORITY MES. NEXT ARRIVAL TIME EQ PP(I,K,1)
550
93

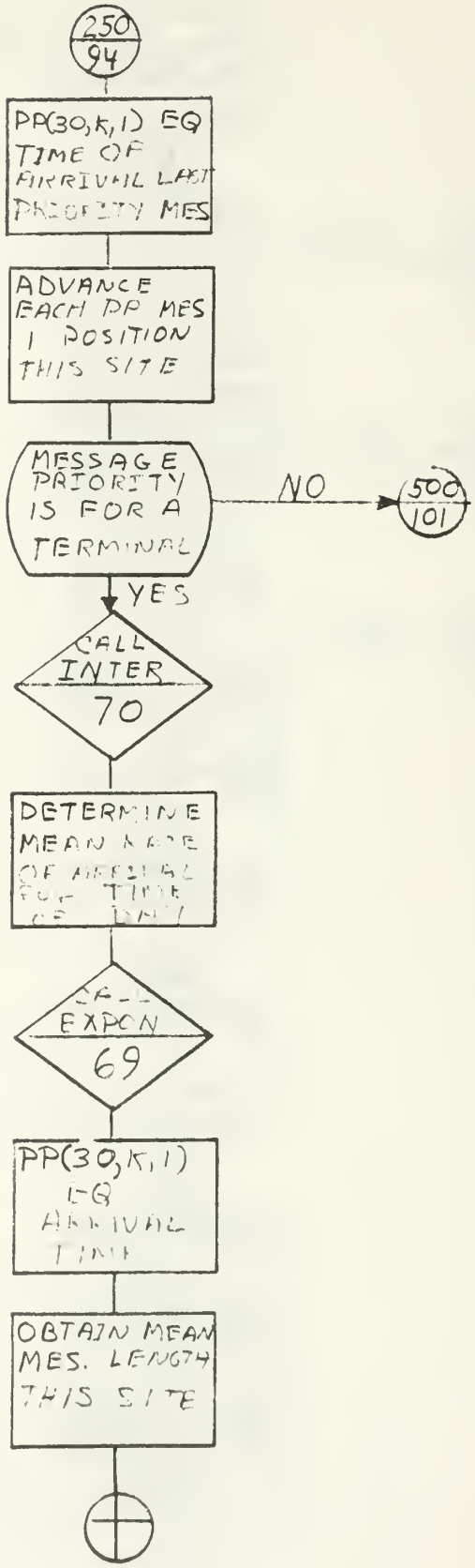

360
92
IMMEDIATE MES. NEXT ARRIVAL TIME EQ OO(I,K,1)
550
93

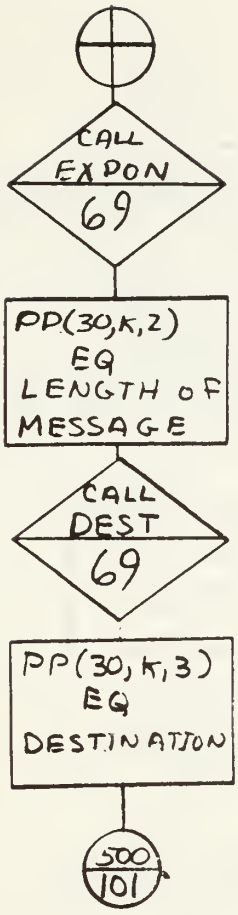

460
92
FLASH MES. NEXT ARRIVAL TIME EQ FF(I,K,1)
550
93

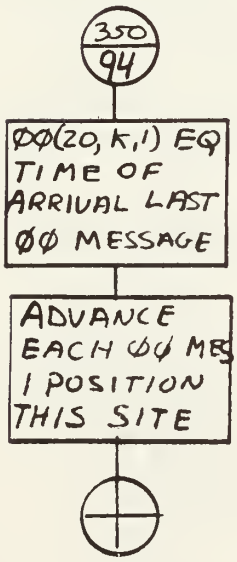

550
93
RETURN MAIN
120
76

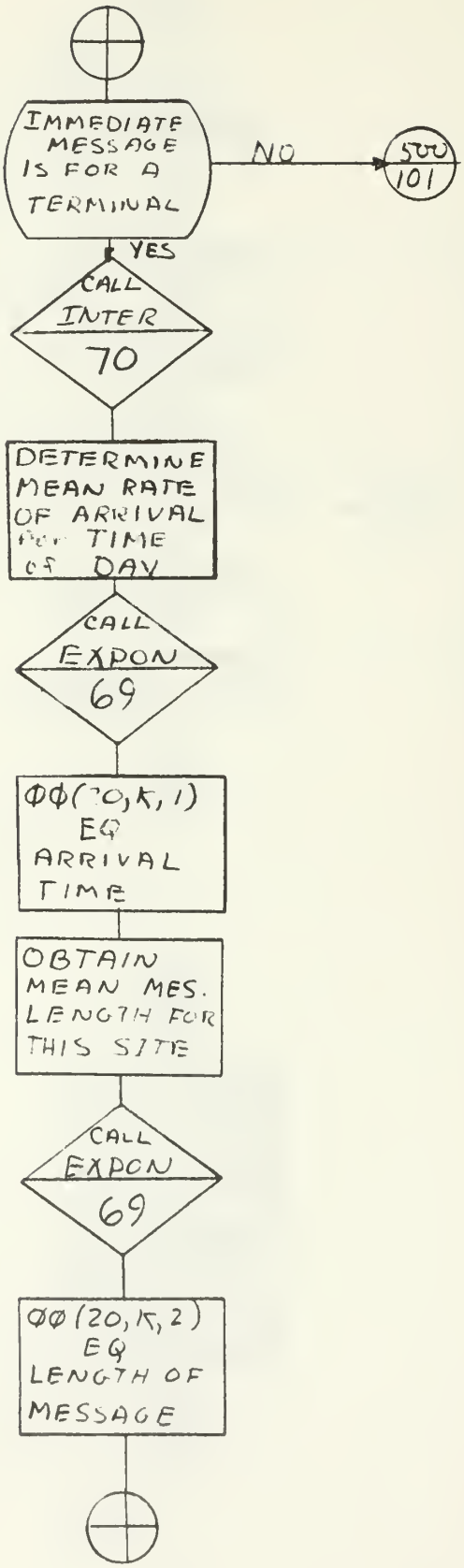

SR
RFILL
71
MESSAGE HAS ROUTINE PRECEDENCE
YES
150
94
NO
MESSAGE HAS PRIORITY PRECEDENCE
YES
250
96
NO
MESSAGE HAS IMMEDIATE PRECEDENCE
YES
350
97
NO
MESSAGE HAS FLASH PRECEDENCE
450
99
150
99
RR(60,K,1) EQ TIME OF ARRIVAL LAST ROUTINE MES
ADVANCE EACH ROUTINE MES 1 POSITION THIS SITE

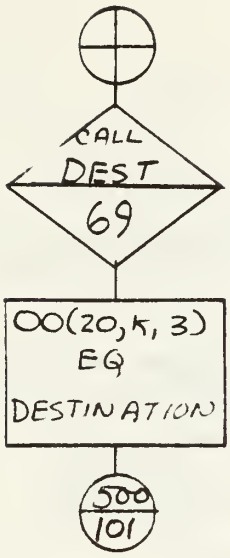

MESSAGE ROUTINE IS FOR A TERMINAL
NO
500
101
YES
CALL INTER
70
DETERMINE MEAN RATE OF ARRIVAL FOR TIME OF DAY
CALL EXPON
69
RR(60,K,1) EQ ARRIVAL TIME
CALL EXPON
69
RR(60,K,2) EQ LENGTH OF MESSAGE
CALL DEST
69
RR(60,K,3) EQ DESTINATION
500
101

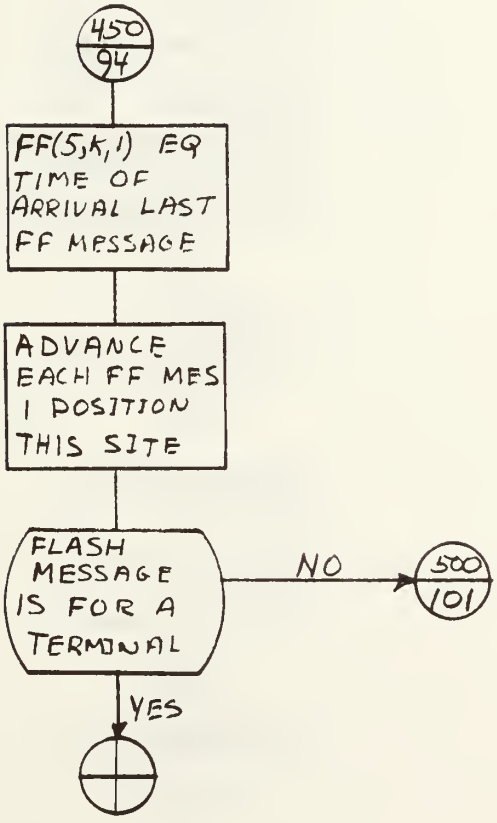

250
94
PP(30,K,1) EQ TIME OF ARRIVAL LAST PRIORITY MES
ADVANCE EACH PP MES 1 POSITION THIS SITE
MESSAGE PRIORITY IS FOR A TERMINAL
NO
500
101
YES
CALL INTER
70
DETERMINE MEAN RATE OF ARRIVAL FOR THIS TYPE OF MES
CALL EXPON
69
PP(30,K,1) EQ ARRIVAL TIME
OBTAIN MEAN MES. LENGTH THIS SITE

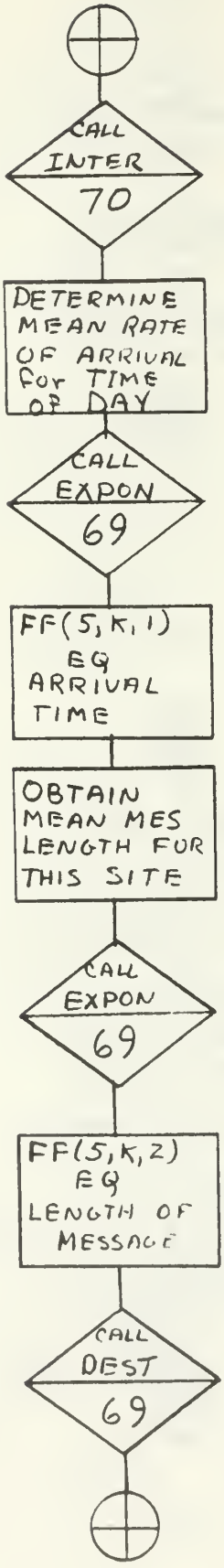

CALL
EXPON
69
PP(30,K,2)
EQ
LENGTH OF
MESSAGE
CALL
DEST
69
PP(30,K,3)
EQ
DESTINATION
500
101

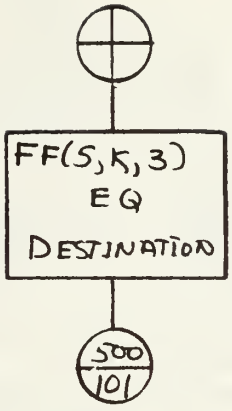

350
94
ØØ(20,K,1) EQ
TIME OF
ARRIVAL LAST
ØØ MESSAGE
ADVANCE
EACH ØØ MES
1 POSITION
THIS SITE

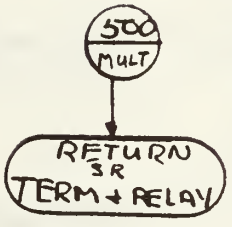

IMMEDIATE MESSAGE IS FOR A TERMINAL
NO
500/101
YES
CALL INTER 70
DETERMINE MEAN RATE OF ARRIVAL FOR TIME OF DAY
CALL EXPON 69
ΦΦ(20,K,1) EQ ARRIVAL TIME
OBTAIN MEAN MES. LENGTH FOR THIS SITE
CALL EXPON 69
ΦΦ(20,K,2) EQ LENGTH OF MESSAGE

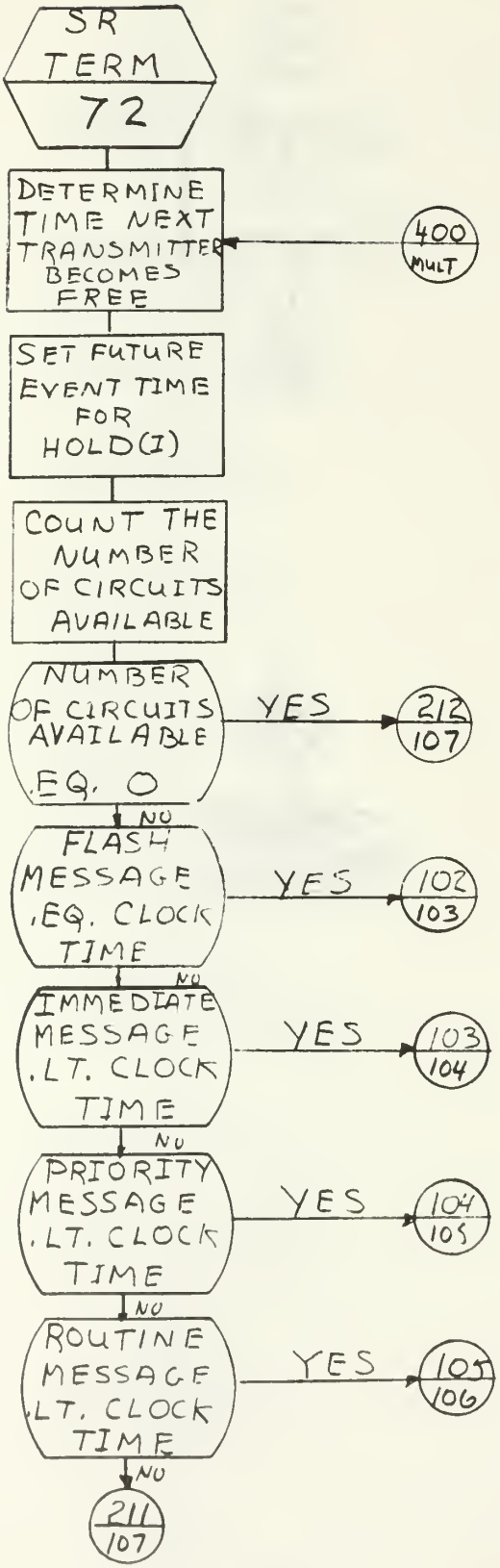

CALL
DEST
69
OO(20,K,3)
EQ
DESTINATION
500
101


450
94
FF(5,K,1) EQ
TIME OF
ARRIVAL LAST
FF MESSAGE
ADVANCE
EACH FF MES
1 POSITION
THIS SITE
FLASH
MESSAGE
IS FOR A
TERMINAL
NO
500
101
YES

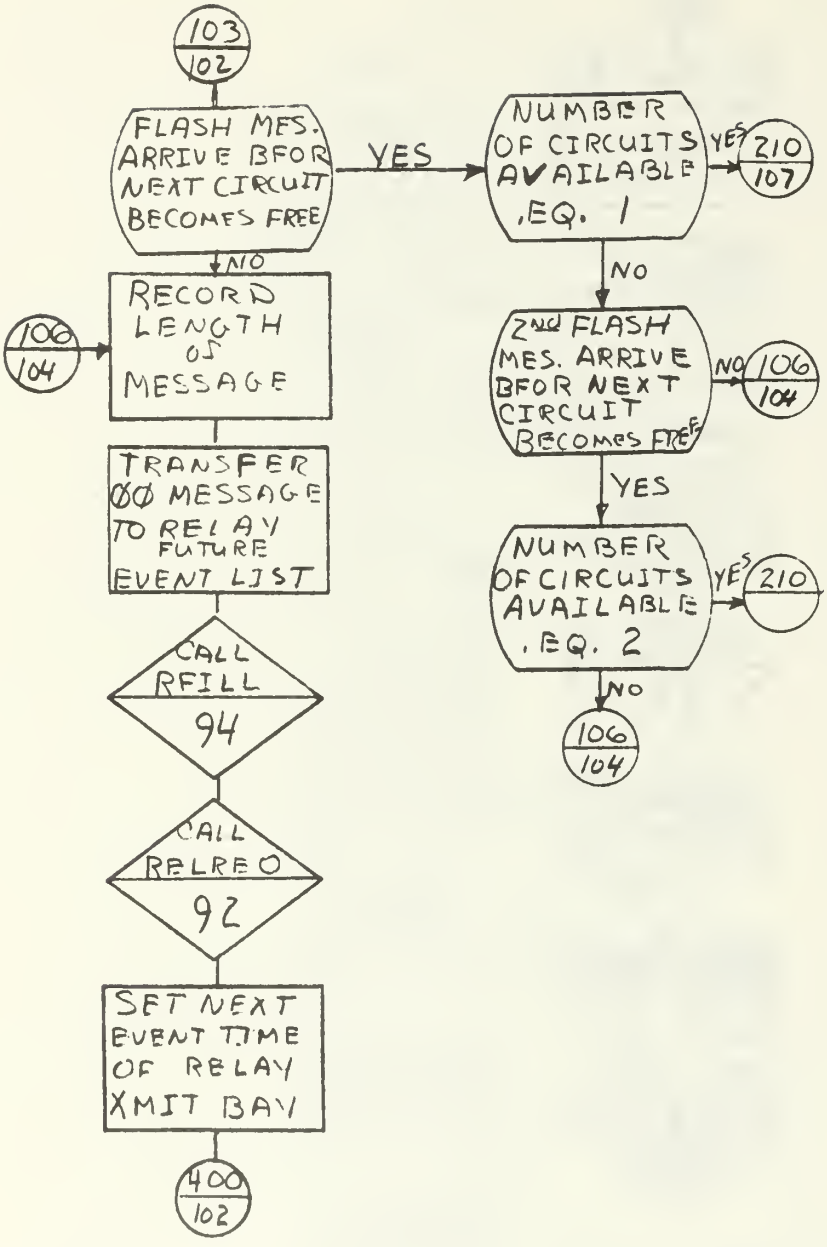

CALL
INTER
70
DETERMINE
MEAN RATE
OF ARRIVAL
For TIME
OF DAY
CALL
EXPON
69
FF(5,K,1)
EQ
ARRIVAL
TIME
OBTAIN
MEAN MES
LENGTH FOR
THIS SITE
CALL
EXPON
69
FF(5,K,2)
EQ
LENGTH OF
MESSAGE
CALL
DEST
69

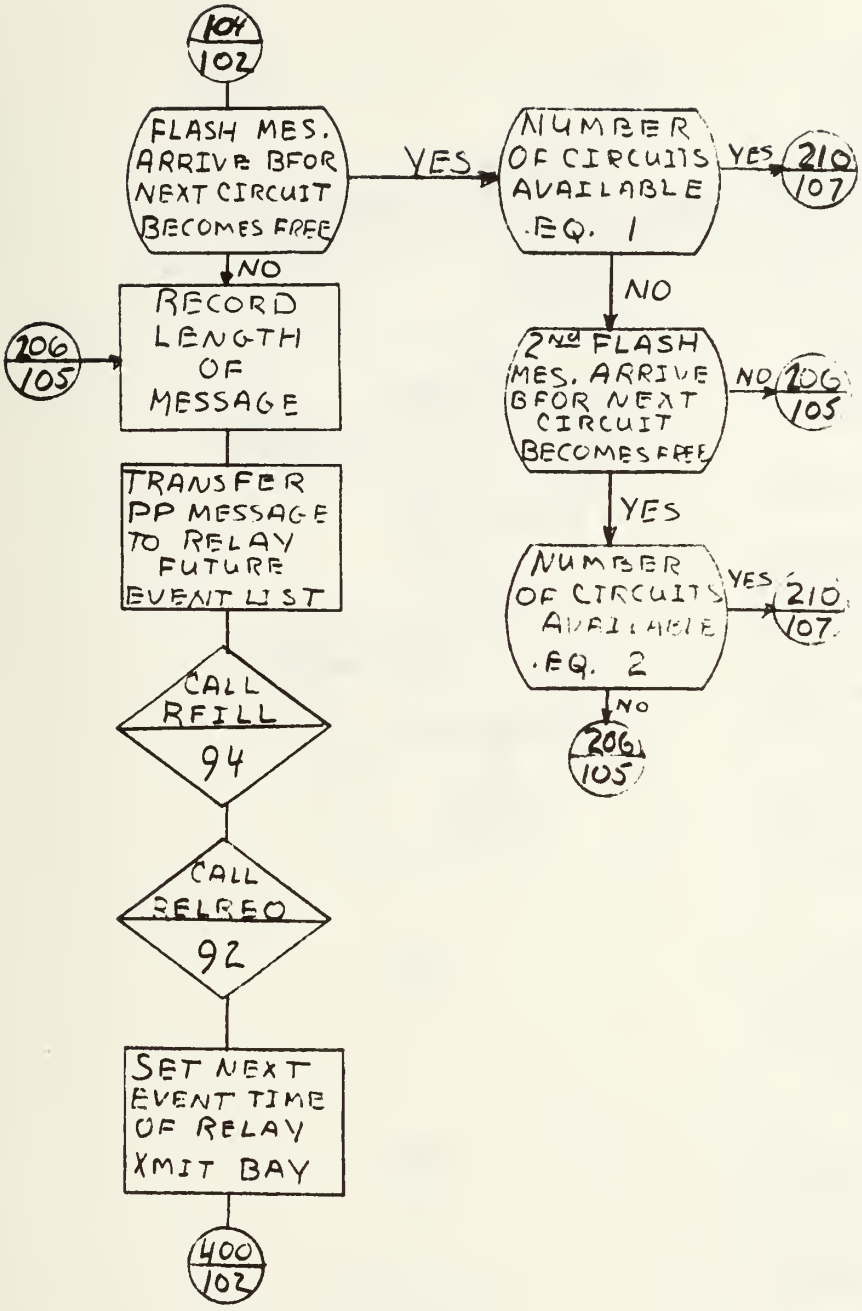

FF(S,K,3)
EQ
DESTINATION
500
101

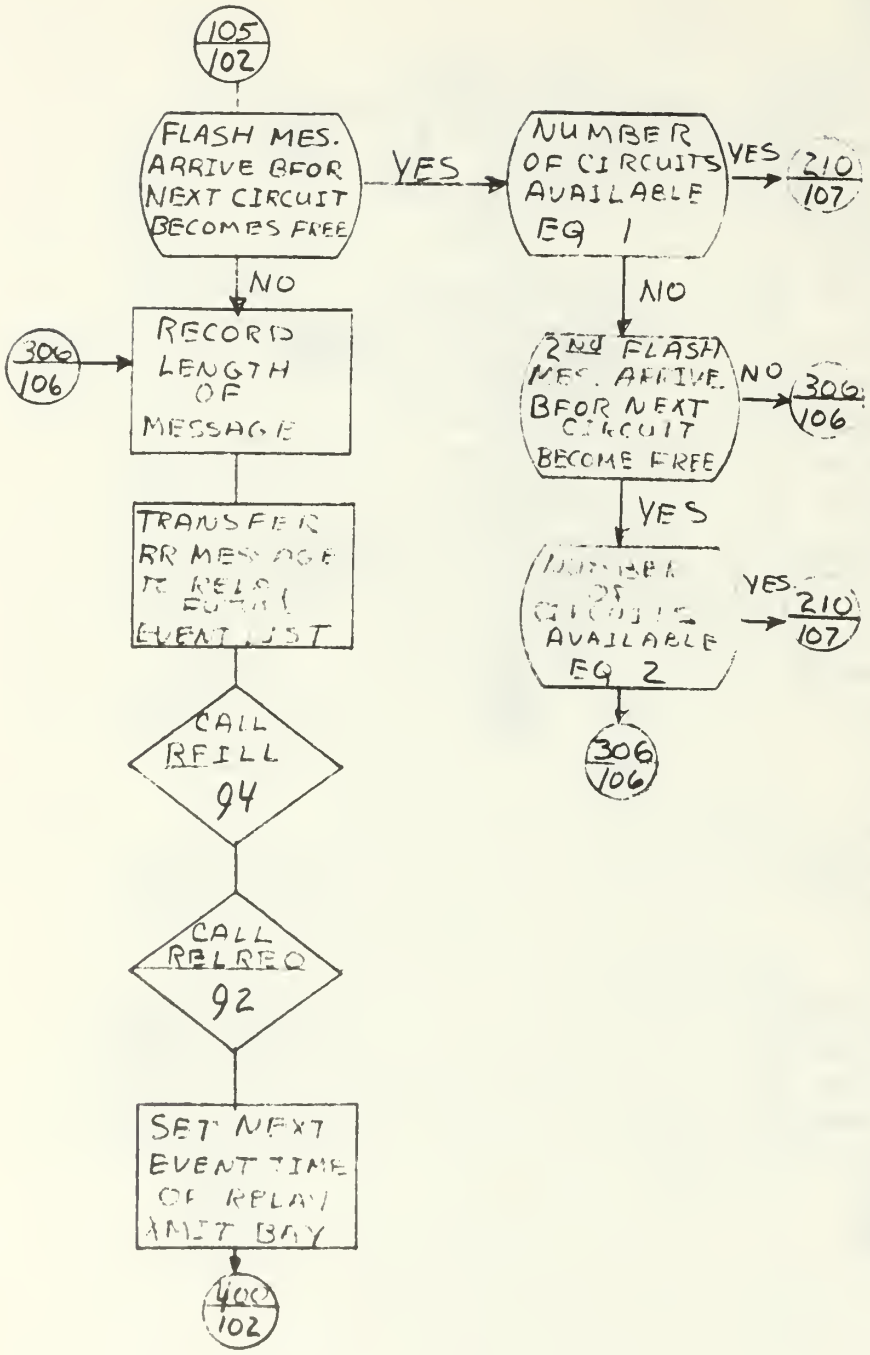

500
MULT
RETURN
3R
TERM + RELAY

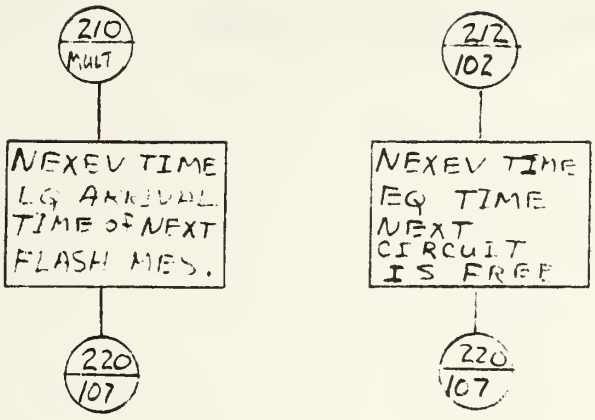

SR
TERM
72
DETERMINE TIME NEXT TRANSMITTER BECOMES FREE
400
MULT
SET FUTURE EVENT TIME FOR HOLD(I)
COUNT THE NUMBER OF CIRCUITS AVAILABLE
NUMBER OF CIRCUITS AVAILABLE .EQ. 0
YES
212
107
NO
FLASH MESSAGE .EQ. CLOCK TIME
YES
102
103
NO
IMMEDIATE MESSAGE .LT. CLOCK TIME
YES
103
104
NO
PRIORITY MESSAGE .LT. CLOCK TIME
YES
104
105
NO
ROUTINE MESSAGE .LT. CLOCK TIME
YES
105
106
NO
211
107

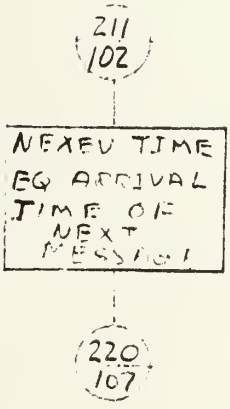

102
102
RESERVE A XMTR FOR MESSAGE LENGTH
RECORD LENGTH OF MESSAGE
TRANSFER MESSAGE TO RELAY FUTURE EVENT LIST
CALL RFILL
94
CALL RELREQ
92
SET NEXT EVENT TIME OF RELAY XMIT BAY
400
102

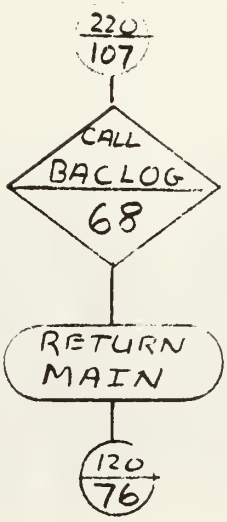

103
102
FLASH MES. ARRIVE BFOR NEXT CIRCUIT BECOMES FREE
YES
NUMBER OF CIRCUITS AVAILABLE .EQ. 1
YES
210
107
NO
NO
RECORD LENGTH OF MESSAGE
106
104
2ND FLASH MES. ARRIVE BFOR NEXT CIRCUIT BECOMES FREE
NO
106
104
YES
TRANSFER ØØ MESSAGE TO RELAY FUTURE EVENT LIST
NUMBER OF CIRCUITS AVAILABLE .EQ. 2
YES
210
CALL RFILL 94
NO
106
104
CALL RELREQ 92
SET NEXT EVENT TIME OF RELAY XMIT BAY
400
102


104
102
FLASH MES. ARRIVE BFOR NEXT CIRCUIT BECOMES FREE
YES
NUMBER OF CIRCUITS AVAILABLE .EQ. 1
YES
210
107
NO
NO
206
105
RECORD LENGTH OF MESSAGE
2ND FLASH MES. ARRIVE BFOR NEXT CIRCUIT BECOMES FREE
NO
206
105
TRANSFER PP MESSAGE TO RELAY FUTURE EVENT LIST
YES
NUMBER OF CIRCUITS AVAILABLE .EQ. 2
YES
210
107
CALL RFILL
94
NO
206
105
CALL RELREQ
92
SET NEXT EVENT TIME OF RELAY XMIT BAY
400
102


105/102
FLASH MES. ARRIVE BFOR NEXT CIRCUIT BECOMES FREE
YES
NUMBER OF CIRCUITS AVAILABLE EQ 1
YES
210/107
NO
NO
306/106
RECORD LENGTH OF MESSAGE
2ND FLASH MES. ARRIVE BFOR NEXT CIRCUIT BECOME FREE
NO
306/106
YES
TRANSFER RR MESSAGE TO RELAY EQUAL EVENT LIST
NUMBER OF CIRCUITS AVAILABLE EQ 2
YES
210/107
306/106
CALL REILL 94
CALL RELREQ 92
SET NEXT EVENT TIME OF RELAY AMIT BAY
400/102


210
MULT
NEXEV TIME
LG ARRIVAL
TIME OF NEXT
FLASH MES.
220
107
212
102
NEXEV TIME
EQ TIME
NEXT
CIRCUIT
IS FREE
220
107


211
102
NEXEV TIME
EQ ARRIVAL
TIME OF
NEXT
MESSAGE
220
107


220
107
CALL
BACLOG
68
RETURN
MAIN
120
76

```
C************** TORN TAPE RELAY NETWORK PROBLEM ****************
C
      IMPLICIT INTEGER (A-W)
      REAL ALOG
      COMMON/C1/ARTE,MMLE,ZPTT
       COMMON/C2/RR,PP,OO,FF
       COMMON/C3/ZLX,ZLY
      COMMON/C4/NEXEV,CLOCK
      COMMON/C5/BACKLP,BACKHP,CRKTS,PUTIL
      COMMON/C6/IX
      COMMON/C7/H1,H2,H3,H4,H5,H6,H7,H8,H9,H10
      COMMON/C8/DIFFCL
      DIMENSION ZLX(8,5),ZLY(8,5),ZPTT(5,5),NEXEV(20,2),
     1HOLD1(6),HOLD2(6),HOLD3(4),HOLD4(5),HOLD5(5),HOLD6(6),
     1HOLD7(5),HOLD8(5),HOLD9(7),HOLD10(4),
     1ARTE(4,5),MMLE(4,5),RR(60,10,3),PP(30,10,3),OO(20,10,3),
     1FF(5,10,3),BACKLP(10),BACKHP(10),CRKTS(10),PUTIL(10)
C     INITIALIZE
C
      COUNT=0
      EVENT=0
      DO 101 K=1,3
      DO 101 J=6,10
      DO 101 I=1,60
       RR(I,J,K)=9000000
101   CONTINUE
C
      DO 102 K=1,3
      DO 102 J=6,10
      DO 102 I=1,30
      PP(I,J,K)=9000000
102   CONTINUE
C
      DO 103 K=1,3
      DO 103 J=6,10
      DO 103 I=1,20
      OO(I,J,K)=9000000
103   CONTINUE
C
      DO 104 K=1,3
      DO 104 J=6,10
      DO 104 I=1,5
      FF(I,J,K)=9000000
104   CONTINUE
C
      DO 301 I=1,H1
301   HOLD1(I)=9000000
      DO 302 I=1,H2
```

```
302     HOLD2(I)=9000000
        DO 303 I=1,H3
303     HOLD3(I)=9000000
        DO 304 I=1,H4
304     HOLD4(I)=9000000
        DO 305 I=1,H5
305     HOLD5(I)=9000000
        DO 306 I=1,H6
306     HOLD6(I)=9000000
        DO 307 I=1,H7
307     HOLD7(I)=9000000
        DO 308 I=1,H8
308     HOLD8(I)=9000000
        DO 309 I=1,H9
309     HOLD9(I)=9000000
        DO 310 I=1,H10
310     HOLD10(I)=9000000
        CALL FILL(1)
        CALL FILL(2)
        CALL FILL(3)
        CALL FILL(4)
        CALL FILL(5)
        CLOCK=61000
120     TEST=(CLOCK-(COUNT*9000))
        IF(TEST.GE.70000)GO TO 250
C        DETERMINE NEXT EVENT TIME
        CLOCK=8000000
        DO 130 I=1,20
        IF(NEXEV(I,1).GE.CLOCK)GO TO 130
        CLOCK=NEXEV(I,1)
        TRANS=NEXEV(I,2)
        EVENT=I
130     CONTINUE
121     GO TO(201,202,203,204,205,206,207,208,209,210,211,212,
       1213,214,215,216,217,218,219,220),TRANS
C
C
C
201     CALL IHOLD(HOLD1,H1,EVENT,&120)
202     CALL IHOLD(HOLD2,H2,EVENT,&120)
203     CALL IHOLD(HOLD3,H3,EVENT,&120)
204     CALL IHOLD(HOLD4,H4,EVENT,&120)
205     CALL IHOLD(HOLD5,H5,EVENT,&120)
206     CALL IHOLD(HOLD6,H6,EVENT,&120)
207     CALL IHOLD(HOLD7,H7,EVENT,&120)
208     CALL IHOLD(HOLD8,H8,EVENT,&120)
209     CALL IHOLD(HOLD9,H9,EVENT,&120)
210     CALL IHOLD(HOLD10,H10,EVENT,&120)
```

```
      CALL TERM (HOLD1,1,H1,EVENT,&120)
      CALL TERM (HOLD2,2,H2,EVENT,&120)
      CALL TERM (HOLD3,3,H3,EVENT,&120)
      CALL TERM (HOLD4,4,H4,EVENT,&120)
      CALL TERM(HOLD5,5,H5,EVENT,&120)
      CALL RELAY (HOLD6,6,H6,EVENT,&120)
      CALL RELAY (HOLD7,7,H7,EVENT,&120)
      CALL RELAY (HOLD8,8,H8,EVENT,&120)
      CALL RELAY (HOLD9,9,H9,EVENT,&120)
      CALL RELAY (HOLD10,10,H10,EVENT,&120)
      CALL UTIL(9)
      COUNT=COUNT+1
      WRITE(6,264)
      DO 251 I=1,10
      WRITE(6,260)BACKLP(I),BACKHP(I),PUTIL(I)
      BACKLP(I)=0
      BACKHP(I)=0
      PUTIL(I)=0
  251 CONTINUE
      IF(COUNT.NE.8)GO TO 120
  260 FORMAT(T10,I10,T30,I10,T50,I10//)
  264 FORMAT(T15,'BACKLP',T35,'BACKHP',T55,'PERCENT UTILIZATION
     1 TIMES 1000'/)
      STOP
      END


      SUBROUTINE FILL(K)
      IMPLICIT INTEGER (A-W)
      COMMON/C1/ARTE,MMLE,ZPTT
       COMMON/C2/RR,PP,OO,FF
       COMMON/C3/ZLX,ZLY
      COMMON/C4/NEXEV,CLOCK
      COMMON/C6/IX
      DIMENSION ARTE(4,5),MMLE(4,5),ZPTT(5,5),RR(60,10,3),PP(30,10,3),OO
     1(20,10,3),FF(5,10,3),ZLX(8,5),ZLY(8,5),NEXEV(20,2)
      TIMER=61000
      DO 100 I=1,60
      CALL INTER(K,TIMER,ZNS)
      MEAN=ARTE(1,K)/ZNS
      CALL EXPON(MEAN,ART)
      TIMER=TIMER+ART
      RR(I,K,1)=TIMER
      MEAN=MMLE(1,K)
      CALL EXPON(MEAN,LONG)
      RR(I,K,2)=LONG
       CALL DEST(K,DES)
  100 RR(I,K,3)=DES
```

```
      TIMEP=61000
      DO 101 I=1,30
      CALL INTER(K,TIMEP,ZNS)
      MEAN=ARTE(2,K)/ZNS
      CALL EXPON(MEAN,ART)
      TIMEP=TIMEP+ART
      PP(I,K,1)=TIMEP
      MEAN=MMLE(2,K)
      CALL EXPON(MEAN,LONG)
      PP(I,K,2)=LONG
       CALL DEST(K,DES)
101   PP(I,K,3)=DES
      TIMEO=61000
      DO 102 I=1,20
      CALL INTER(K,TIMEO,ZNS)
      MEAN=ARTE(3,K)/ZNS
      CALL EXPON(MEAN,ART)
      TIMEO=TIMEO+ART
      OO(I,K,1)=TIMEO
      MEAN=MMLE(3,K)
      CALL EXPON(MEAN,LONG)
      OO(I,K,2)=LONG
       CALL DEST(K,DES)
102   OO(I,K,3)=DES
      TIMEF=61000
      DO 103 I=1,5
      CALL INTER(K,TIMEF,ZNS)
      MEAN=ARTE(4,K)/ZNS
      CALL EXPON(MEAN,ART)
      TIMEF=TIMEF+ART
      FF(I,K,1)=TIMEF
      MEAN=MMLE(4,K)
      CALL EXPON(MEAN,LONG)
      FF(I,K,2)=LONG
       CALL DEST(K,DES)
103   FF(I,K,3)=DES
      K10=K+10
      NEXEV(K10,1)=MINC(RR(1,K,1),PP(1,K,1),OO(1,K,1),FF(1,K,1))
      RETURN
      END


      SUBROUTINE RANDO(IY,YFL)
      COMMON/C6/IX
      IY=IX*65539
      IF(IY)5,6,6
5     IY=IY+2147483647+1
6     YFL=IY
```

```
      YFL=YFL*.4656613E-9
      RETURN
      END


      SUBROUTINE EXPON(MEAN,VAR)
      IMPLICIT INTEGER (A-W)
      REAL ALOG
      COMMON/C6/IX
      CALL RANDO(IY,YFL)
      IX=IY
      VAR=-MEAN*ALOG(YFL)
      RETURN
      END


      SUBROUTINE INTER(K,TIME,ZNS)
      IMPLICIT INTEGER (A-W)
      COMMON/C3/ZLX,ZLY
      DIMENSION ZLX(8,5),ZLY(8,5)
C
      TEMP=TIME
      SPEED=TIME
      IF(TEMP.LE.70000)GO TO 500
      CHECK=(TEMP-61000)/9000
      TEMP=TEMP-(9000*CHECK)
500   I=1
100   IF(ZLX(I,K)-TEMP)30,40,50
30    I=I+1
      GO TO 100
40    ZNS=ZLY(I,K)
      GO TO 200
50    IM1=I-1
      ZNS=ZLY(IM1,K)+(ZLY(I,K)-ZLY(IM1,K))/(ZLX(I,K)-ZLX
     1(IM1,K))*(TEMP-ZLX(IM1,K))
200   RETURN
      END


      SUBROUTINE DEST(J,DES)
      IMPLICIT INTEGER (A-W)
      COMMON/C1/ARTE,MMLE,ZPTT
      COMMON/C6/IX
      DIMENSION ARTE(4,5),MMLE(4,5),ZPTT(5,5)
      CALL RANDO(IY,YFL)
      IX=IY
      I=1
100   IF(ZPTT(I,J)-YFL)30,40,40
```

```
40      DES=I
        GO TO 200
30      I=I+1
        GO TO 100
200     RETURN
        END


        SUBROUTINE IHOLD(HOLD,L,M,*)
        IMPLICIT INTEGER (A-W)
        COMMON/C4/NEXEV,CLOCK
        DIMENSION HOLD(L)          ,NEXEV(20,2)
        DO 101 I=1,L
            IF(HOLD(I).NE.NEXEV(M,1)) GO TO 101
        HOLD(I)=9000000
101     CONTINUE
C
        L1=L-1
        DO 301 I=1,L1
        IP1=I+1
        DO 301 J=IP1,L
        IF(HOLD(I).LE.HOLD(J))GO TO 301
        TEMP=HOLD(I)
        HOLD(I)=HOLD(J)
        HOLD(J)=TEMP
301     CONTINUE
        NEXEV(M,1)=HOLD(1)
C
        RETURN 1
        END


        SUBROUTINE TERM(HOLD,K,L,M,*)
        IMPLICIT INTEGER (A-W)
        REAL ALOG
         COMMON/C2/RR,PP,QQ,FF
        COMMON/C4/NEXEV,CLOCK
        COMMON/C5/BACKLP,BACKHP,CRKTS,PUTIL
        COMMON/C8/DIFFCL
        DIMENSION RR(60,10,3),PP(30,10,3),QQ(20,10,3),FF(5,10,3),HOLD(L),
       1NEXEV(20,2),BACKLP(10),BACKHP(10),CRKTS(10),
       1PUTIL(10)
        L1=L-1
400     DO 100 I=1,L1
        IP1=I+1
        DO 100 J=IP1,L
        IF(HOLD(I).LE.HOLD(J))GO TO 100
        TEMP=HOLD(I)
```

```
      HOLD(I)=HOLD(J)
      HOLD(J)=TEMP
100   CONTINUE
C   PUT TIME NEXT CIRCUIT FREE IN POSITION ONE
C     AND SET FUTURE CLOCK TIME FOR HOLD(I)
C
      NEXEV(K,1)=HOLD(1)
      AVAIL=0
      DO 101 I=1,L
      IF(HOLD(I).NE.9000000)GO TO 101
      AVAIL=AVAIL+1
101   CONTINUE
C
C   COUNT THE NUMBER OF CIRCUITS AVAILABLE
701   IF(AVAIL.EQ.0)GO TO 212
C
      IF(FF(1,K,1).EQ.NEXEV(M,1))GO TO 102
      IF(OO(1,K,1).LE.NEXEV(M,1))GO TO 103
      IF(PP(1,K,1).LE.NEXEV(M,1)) GO TO 104
      IF(RR(1,K,1).LE.NEXEV(M,1)) GO TO 105
      GO TO 211
C
102   HOLD(L)=FF(1,K,1)+FF(1,K,2)
      DEST=FF(1,K,3)+5
      PUTIL(K)=PUTIL(K)+FF(1,K,2)
      FF(5,DEST,1)=NEXEV(M,1)+FF(1,K,2)
      FF(5,DEST,2)=FF(1,K,2)
      CALL RFILL(K,2,4)
      CALL RELREO(DEST,4)
      RLAYT=DEST+10
      IF(FF(1,DEST,1).EQ.NEXEV(RLAYT,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
      IF(NEXEV(RLAYT,1).EQ.NEXEV(DEST,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
      NEXEV(RLAYT,1)=MINO(RR(1,DEST,1),PP(1,DEST,1),
     100(1,DEST,1),FF(1,DEST,1))
      GO TO 400
103   IF(FF(1,K,1).LE.HOLD(1).AND.HOLD(1).NE.9000000)GO TO 201
106   HOLD(L)=NEXEV(M,1)+OO(1,K,2)
      DEST=OO(1,K,3)+5
      PUTIL(K)=PUTIL(K)+OO(1,K,2)
      OO(20,DEST,1)=NEXEV(M,1)+OO(1,K,2)
      OO(20,DEST,2)=OO(1,K,2)
      CALL RFILL(K,2,3)
      CALL RELREO(DEST,3)
      RLAYT=DEST+10
      IF(FF(1,DEST,1).EQ.NEXEV(RLAYT,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
```

```
      IF(NEXEV(RLAYT,1).EQ.NEXEV(DEST,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
      NEXEV(RLAYT,1)=MIN0(RR(1,DEST,1),PP(1,DEST,1),
     100(1,DEST,1),FF(1,DEST,1))
      GO TO 400
C
201   IF(AVAIL.EQ.1)GO TO 210
      IF(FF(2,K,1).LE.HOLD(1))GO TO 202
      GO TO 106
202   IF(AVAIL.EQ.2)GO TO 210
      GO TO 106
C
104   IF(FF(1,K,1).LE.HOLD(1).AND.HOLD(1).NE.9000000)GO TO 301
206   HOLD(L)=NEXEV(M,1)+PP(1,K,2)
      DEST=PP(1,K,3)+5
      PUTIL(K)=PUTIL(K)+PP(1,K,2)
      PP(30,DEST,1)=NEXEV(M,1)+PP(1,K,2)
      PP(30,DEST,2)=PP(1,K,2)
      CALL RFILL(K,2,2)
      CALL RELREQ(DEST,2)
      RLAYT=DEST+10
      IF(FF(1,DEST,1).EQ.NEXEV(RLAYT,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
      IF(NEXEV(RLAYT,1).EQ.NEXEV(DEST,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
      NEXEV(RLAYT,1)=MIN0(RR(1,DEST,1),PP(1,DEST,1),
     100(1,DEST,1),FF(1,DEST,1))
      GO TO 400
C
301   IF(AVAIL.EQ.1)GO TO 210
      IF(FF(2,K,1).LE.HOLD(1))GO TO 302
      GO TO 206
302   IF(AVAIL.EQ.2)GO TO 210
      GO TO 206
C
105   IF(FF(1,K,1).LE.HOLD(1).AND.HOLD(1).NE.9000000)GO TO 401
306   HOLD(L)=NEXEV(M,1)+RR(1,K,2)
      DEST=RR(1,K,3)+5
      PUTIL(K)=PUTIL(K)+RR(1,K,2)
      RR(60,DEST,1)=NEXEV(M,1)+RR(1,K,2)
      RR(60,DEST,2)=RR(1,K,2)
      CALL RFILL(K,2,1)
      CALL RELREQ(DEST,1)
      RLAYT=DEST+10
      IF(FF(1,DEST,1).EQ.NEXEV(RLAYT,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
      IF(NEXEV(RLAYT,1).EQ.NEXEV(DEST,1).AND.NEXEV(RLAYT,1)
     1.NE.9000000)GO TO 400
```

```
      NEXEV(RLAYT,1)=MINO(RR(1,DEST,1),PP(1,DEST,1),
     100(1,DEST,1),FF(1,DEST,1))
      GO TO 400
C
401   IF(AVAIL.EQ.1)GO TO 210
      IF(FF(2,K,1).LE.HOLD(1))GO TO 402
      GO TO 306
402   IF(AVAIL.EQ.2)GO TO 210
      GO TO 306
C
210   PUTIL(K)=PUTIL(K)+(FF(1,K,1)-NEXEV(M,1))
      NEXEV(M,1)=FF(1,K,1)
      GO TO 220
211   NEXEV(M,1)=MINO(FF(1,K,1),OO(1,K,1),PP(1,K,1),RR(1,K,1))
      GO TO 220
212   NEXEV(M,1)=HOLD(1)
C
220   DIFFCL=0
      CALL BACLOG(K)
999   RETURN 1
      END


      SUBROUTINE RELAY(HOLD,K,L,M,*)
      IMPLICIT INTEGER (A-W)
      REAL ALOG
       COMMON/C2/RR,PP,OO,FF
      COMMON/C4/NEXEV,CLOCK
      COMMON/C5/BACKLP,BACKHP,CRKTS,PUTIL
      COMMON/C8/DIFFCL
      DIMENSION RR(60,10,3),PP(30,10,3),OO(20,10,3),FF(5,10,3),HOLD(L),
     1NEXEV(20,2),BACKLP(10),BACKHP(10),CRKTS(10),
     1PUTIL(10)
      L1=L-1
400   DO 100 I=1,L1
      IP1=I+1
      DO 100 J=IP1,L
      IF(HOLD(I).LE.HOLD(J))GO TO 100
      TEMP=HOLD(I)
      HOLD(I)=HOLD(J)
      HOLD(J)=TEMP
100   CONTINUE
C    PUT TIME NEXT CIRCUIT FREE IN POSITION ONE
C      AND SET FUTURE CLOCK TIME FOR HOLD(I)
      NEXEV(K,1)=HOLD(1)
C
      AVAIL=0
      DO 101 I=1,L
```

```
      IF(HOLD(I).NE.9000000)GO TO 101
      AVAIL=AVAIL+1
101   CONTINUE
C    COUNT THE NUMBER OF CIRCUITS AVAILABLE
701   IF(AVAIL.EQ.0)GO TO 212
C
      IF(FF(1,K,1).EQ.NEXEV(M,1).AND.FF(1,K,1) .NE.9000000)GO TO 102
      IF(OO(1,K,1).LE.NEXEV(M,1).AND.OO(1,K,1) .NE.9000000)GO TO 103
      IF(PP(1,K,1).LE.NEXEV(M,1).AND.PP(1,K,1) .NE.9000000)GO TO 104
      IF(RR(1,K,1).LE.NEXEV(M,1).AND.RR(1,K,1) .NE.9000000)GO TO 105
      GO TO 211
C
102   HOLD(L)=FF(1,K,1)+FF(1,K,2)
      PUTIL(K)=PUTIL(K)+FF(1,K,2)
      FF(1,K,1)=9000000
      CALL REILL(K,1,4)
      GO TO 400
103   IF(FF(1,K,1).LE.HOLD(1).AND.HOLD(1).NE.9000000)GO TO 201
106   HOLD(L)=NEXEV(M,1)+OO(1,K,2)
      PUTIL(K)=PUTIL(K)+OO(1,K,2)
      OO(1,K,1)=9000000
      CALL REILL(K,1,3)
      GO TO 400
C
201   IF(AVAIL.EQ.1)GO TO 210
      IF(FF(2,K,1).LE.HOLD(1))GO TO 202
      GO TO 106
202   IF(AVAIL.EQ.2)GO TO 210
      GO TO 106
C
104   IF(FF(1,K,1).LE.HOLD(1).AND.HOLD(1).NE.9000000)GO TO 301
206   HOLD(L)=NEXEV(M,1)+PP(1,K,2)
      PUTIL(K)=PUTIL(K)+PP(1,K,2)
      PP(1,K,1)=9000000
      CALL REILL(K,1,2)
      GO TO 400
C
301   IF(AVAIL.EQ.1)GO TO 210
      IF(FF(2,K,1).LE.HOLD(1))GO TO 302
      GO TO 206
302   IF(AVAIL.EQ.2)GO TO 210
      GO TO 206
C
105   IF(FF(1,K,1).LE.HOLD(1).AND.HOLD(1).NE.9000000)GO TO 401
306   HOLD(L)=NEXEV(M,1)+RR(1,K,2)
      PUTIL(K)=PUTIL(K)+RR(1,K,2)
      CALL REILL(K,1,1)
      GO TO 400
```

```
C
401   IF(AVAIL.EQ.1)GO TO 210
      IF(FF(2,K,1).LE.HOLD(1))GO TO 402
      GO TO 306
402   IF(AVAIL.EQ.2)GO TO 210
      GO TO 306
C
210   PUTIL(K)=PUTIL(K)+(FF(1,K,1)-NEXEV(M,1))
      NEXEV(M,1)=FF(1,K,1)
      GO TO 220
211   NEXEV(M,1)=MINO(FF(1,K,1),OO(1,K,1),PP(1,K,1),RR(1,K,1))
      GO TO 220
212   NEXEV(M,1)=HOLD(1)
C
220   DIFFCL=0
      CALL BACLOG(K)
      RETURN 1
      END


      SUBROUTINE RFILL (K,MOVE,PREC)
      IMPLICIT INTEGER (A-W)
      COMMON/C1/ARTE,MMLE,ZPTT
       COMMON/C2/RR,PP,OO,FF
       COMMON/C3/ZLX,ZLY
      DIMENSION ARTE(4,5),MMLE(4,5),ZPTT(5,5),RR(60,10,3),PP(30,10,3),OO
     1(20,10,3),FF(5,10,3),ZLX(8,5),ZLY(8,5)
      IF(PREC.EQ.1)GO TO 100
      IF(PREC.EQ.2)GO TO 200
      IF(PREC.EQ.3) GO TO 300
      IF(PREC.EQ.4)GO TO 400
100   TIME=RR(60,K,1)
      DO 101 I=1,59
      IP1=I+1
      RR(I,K,1)=RR(IP1,K,1)
      RR(I,K,2)=RR(IP1,K,2)
      RR(I,K,3)=RR(IP1,K,3)
101   CONTINUE
      IF(MOVE.EQ.1)GO TO 500
C
      CALL INTER(K,TIME,ZNS)
      MEAN=ARTE(1,K)/ZNS
      CALL EXPON(MEAN,ART)
      RR(60,K,1)=TIME+ART
      MEAN=MMLE(1,K)
      CALL EXPON(MEAN,LONG)
      RR(60,K,2)=LONG
       CALL DEST(K,DES)
```

```
      RR(60,K,3)=DES
      GO TO 500
  200 TIME=PP(30,K,1)
      DO 201 I=1,29
      IP1=I+1
      PP(I,K,1)=PP(IP1,K,1)
      PP(I,K,2)=PP(IP1,K,2)
      PP(I,K,3)=PP(IP1,K,3)
  201 CONTINUE
      IF(MOVE.EQ.1)GO TO 500
C
      CALL INTER(K,TIME,ZNS)
         MEAN=ARTE(2,K)/ZNS
      CALL EXPON(MEAN,ART)
      PP(30,K,1)=TIME+ART
      MEAN=MMLE(2,K)
      CALL EXPON(MEAN,LONG)
      PP(30,K,2)=LONG
       CALL DEST(K,DES)
      PP(30,K,3)=DES
      GO TO 500
C
  300 TIME=OO(20,K,1)
      DO 301 I=1,19
      IP1=I+1
      OO(I,K,1)=OO(IP1,K,1)
      OO(I,K,2)=OO(IP1,K,2)
      OO(I,K,3)=OO(IP1,K,3)
  301 CONTINUE
      IF(MOVE.EQ.1)GO TO 500
C
      CALL INTER(K,TIME,ZNS)
      MEAN=ARTE(3,K)/ZNS
      CALL EXPON(MEAN,ART)
      OO(20,K,1)=TIME+ART
      MEAN=MMLE(3,K)
      CALL EXPON(MEAN,LONG)
      OO(20,K,2)=LONG
       CALL DEST(K,DES)
      OO(20,K,3)=DES
      GO TO 500
C
  400 TIME=FF(5,K,1)
      DO 401 I=1,4
      IP1=I+1
      FF(I,K,1)=FF(IP1,K,1)
      FF(I,K,2)=FF(IP1,K,2)
      FF(I,K,3)=FF(IP1,K,3)
```

```
401    CONTINUE
       IF(MOVE.EQ.1)GO TO 500
       CALL INTER(K,TIME,ZNS)
       MEAN=ARTE(4,K)/ZNS
       CALL EXPON(MEAN,ART)
       FF(5,K,1)=TIME+ART
       MEAN=MMLE(4,K)
       CALL EXPON(MEAN,LONG)
       FF(5,K,2)=LONG
        CALL DEST(K,DES)
       FF(5,K,3)=DES
500    RETURN
       END

       SUBROUTINE RELREQ(K,PREC)
       IMPLICIT INTEGER (A-W)
       COMMON/C2/RR,PP,QQ,FF
       DIMENSION RR(60,10,3),PP(30,10,3),QQ(20,10,3),
      1FF(5,10,3)
       GO TO(100,200,300,400),PREC
100    DO 101 I=1,59
       IP1=I+1
       DO 101 J=IP1,60
       IF(RR(I,K,1).LE.RR(J,K,1))GO TO 101
       TEMP1=RR(I,K,1)
       TEMP2=RR(I,K,2)
       TEMP3=RR(I,K,3)
       RR(I,K,1)=RR(J,K,1)
       RR(I,K,2)=RR(J,K,2)
       RR(I,K,3)=RR(J,K,3)
       RR(J,K,1)=TEMP1
       RR(J,K,2)=TEMP2
       RR(J,K,3)=TEMP3
101    CONTINUE
       GO TO 500
C
200    DO 201 I=1,29
       IP1=I+1
       DO 201 J=IP1,30
       IF(PP(I,K,1).LE.PP(J,K,1))GO TO 201
       TEMP1=PP(I,K,1)
       TEMP2=PP(I,K,2)
       TEMP3=PP(I,K,3)
       PP(I,K,1)=PP(J,K,1)
       PP(I,K,2)=PP(J,K,2)
       PP(I,K,3)=PP(J,K,3)
       PP(J,K,1)=TEMP1
```

```
      PP(J,K,2)=TEMP2
      PP(J,K,3)=TEMP3
201   CONTINUE
C
300   DO 301 I=1,19
      IP1=I+1
      DO 301 J=IP1,20
      IF(QQ(I,K,1).LE.QQ(J,K,1))GO TO 301
      TEMP1=QQ(I,K,1)
      TEMP2=QQ(I,K,2)
      TEMP3=QQ(I,K,3)
      QQ(I,K,1)=QQ(J,K,1)
      QQ(I,K,2)=QQ(J,K,2)
      QQ(I,K,3)=QQ(J,K,3)
      QQ(J,K,1)=TEMP1
      QQ(J,K,2)=TEMP2
      QQ(J,K,3)=TEMP3
301   CONTINUE
C
400   DO 401 I=1,4
      IP1=I+1
      DO 401 J=IP1,5
      IF(FF(I,K,1).LE.FF(J,K,1))GO TO 401
      TEMP1=FF(I,K,1)
      TEMP2=FF(I,K,2)
      TEMP3=FF(I,K,3)
      FF(I,K,1)=FF(J,K,1)
      FF(I,K,2)=FF(J,K,2)
      FF(I,K,3)=FF(J,K,3)
      FF(J,K,1)=TEMP1
      FF(J,K,2)=TEMP2
      FF(J,K,3)=TEMP3
401   CONTINUE
500   RETURN
      END


      SUBROUTINE BACLOG(K)
      IMPLICIT INTEGER (A-W)
      REAL ALOG
      COMMON/C2/RR,PP,QQ,FF
      COMMON/C4/NEXEV,CLOCK
      COMMON/C5/BACKLP,BACKHP,CRKTS,PUTIL
      COMMON/C8/DIFFCL
      DIMENSION RR(60,10,3),PP(30,10,3),QQ(20,10,3),
     1FF(5,10,3),NEXEV(20,2),BACKLP(10),BACKHP(10),CRKTS(10),
     1PUTIL(10)
C
```

```
      BACKL=DIFFCL
      I=1
100   IF(CLOCK .LT.RR(I,K,1))GO TO 101
      BACKL=BACKL+RR(I,K,2)
      I=I+1
      GO TO 100
101   I=1
302   IF(CLOCK .LT.PP(I,K,1))GO TO 102
      BACKL=BACKL+PP(I,K,2)
      I=I+1
      GO TO 302
102   BACKL=BACKL/(CRKTS(K)*60)
      BACKLP(K)=MAXO(BACKLP(K),BACKL)
      BACKH=DIFFCL
      I=1
200   IF(CLOCK.LT.QQ(I,K,1))GO TO 201
      BACKH=BACKH+QQ(I,K,2)
      I=I+1
      GO TO 200
201   I=1
402   IF(CLOCK.LT.FF(I,K,1))GO TO 202
      BACKH=BACKH+FF(I,K,2)
      I=I+1
      GO TO 402
202   BACKH=BACKH/(CRKTS(K)*60)
      BACKHP(K)=MAXO(BACKHP(K),BACKH)
      RETURN
      END

      SUBROUTINE UTIL(CLOCK)
      IMPLICIT INTEGER (A-W)
      REAL ALOG
      COMMON/C5/BACKLP,BACKHP,CRKTS,PUTIL
      COMMON/C7/H1,H2,H3,H4,H5,H6,H7,H8,H9,H10
      DIMENSION BACKLP(10),BACKHP(10),CRKTS(10),PUTIL(10)
      PUTIL(1)=PUTIL(1)/(CLOCK*H1)
      PUTIL(2)=PUTIL(2)/(CLOCK*H2)
      PUTIL(3)=PUTIL(3)/(CLOCK*H3)
      PUTIL(4)=PUTIL(4)/(CLOCK*H4)
      PUTIL(5)=PUTIL(5)/(CLOCK*H5)
      PUTIL(6)=PUTIL(6)/(CLOCK*H6)
      PUTIL(7)=PUTIL(7)/(CLOCK*H7)
      PUTIL(8)=PUTIL(8)/(CLOCK*H8)
      PUTIL(9)=PUTIL(9)/(CLOCK*H9)
      PUTIL(10)=PUTIL(10)/(CLOCK*H10)
      RETURN
      END
```

```
C
      BLOCK DATA
      IMPLICIT INTEGER (A-W)
      REAL ALOG
      COMMON/C1/ARTE,MMLE,ZPTT
       COMMON/C3/ZLX,ZLY
      COMMON/C4/NEXEV,CLOCK
      COMMON/C5/BACKLP,BACKHP,CRKTS,PUTIL
      COMMON/C6/IX
      COMMON/C7/H1,H2,H3,H4,H5,H6,H7,H8,H9,H10
      DIMENSION ARTE(4,5),MMLE(4,5),ZPTT(5,5),ZLX(8,5),
     1ZLY(8,5),NEXEV(20,2),BACKLP(10),BACKHP(10),CRKTS(10),
     1PUTIL(10)
      DATA ZLX/0.,21600.,43200.,57600.,72000.,86400.,86400.,
     186400.,0.,14400.,36000.,57600.,64800.,79200.,86400.,
     186400.,0.,28800.,43200.,64800.,72000.,86400.,86400.,86400.,
     10.,14400.,21600.,43200.,50400.,64800.,79800.,86400.,
     10.,21600.,29800.,43200.,57600.,72000.,86400.,86400./
C
      DATA ZLY/.6,.4,.7,1.,1.,.6,.6,.6,
     1.5,.5,.3,.7,1.,1.,.5,.5,
     1.6,.6,1.,1.,.8,.6,.6,.6,
     1.3,.3,.6,.6,1.,1.,.5,.3,
     1.3,.3,.5,.5,1.,1.,.3,.3/
C
      DATA ZPTT/0.,.2,.5,.9,1.,
     1.25,.25,.4,.75,1.,
     1.4,.8,.8,.9,1.,
     1.2,.4,.7,.7,1.,
     1.2,.5,.6,1.,1./
C
      DATA ARTE/30,70,190,1000,40,60,90,500,60,100,130,700,
     150,100,140,900,40,60,100,1000/
C
      DATA MMLE/100,85,60,30,140,90,50,20,120,80,45,25,
     1130,100,60,30,100,80,60,30/
C
      DATA NEXEV/10*9000000,5*0,5*9000000,
     11,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20/
      DATA BACKLP,BACKHP,CRKTS/10*0,10*0,6,6,4,5,5,6,5,5,7,
     14/
      DATA PUTIL/10*0/
      DATA IX/1933/
      DATA H1,H2,H3,H4,H5,H6,H7,H8,H9,H10/6,6,4,5,5,6,5,5,7,
     14/
      END
```

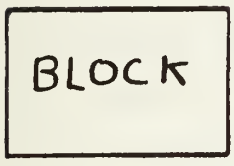
APPENDIX C

GPSS Computer Model

Appendix C contains a GPSS flow chart of the torn tape relay network and the GPSS computer program used to simulate this system.

GPSS FLOW CHART SYMBOLS

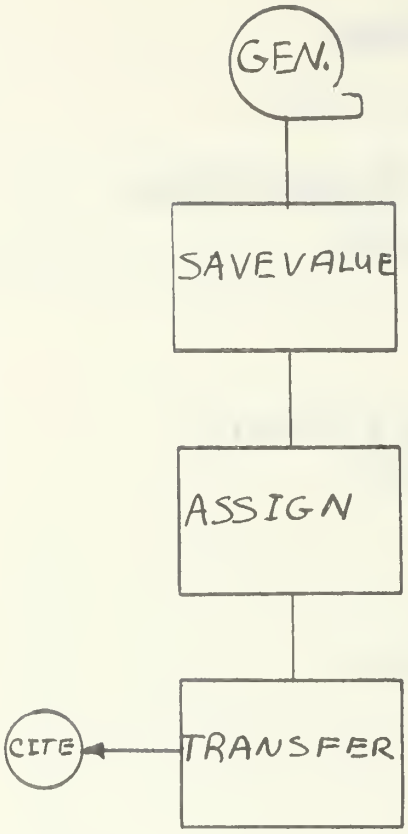
GENERATE
TRANSACTIONS



43 GPSS BLOCK TYPES
(Ref. 6, 7, 8, 14)

X IS A MNEMONIC
CONNECTOR

OFF PAGE
CONNECTOR

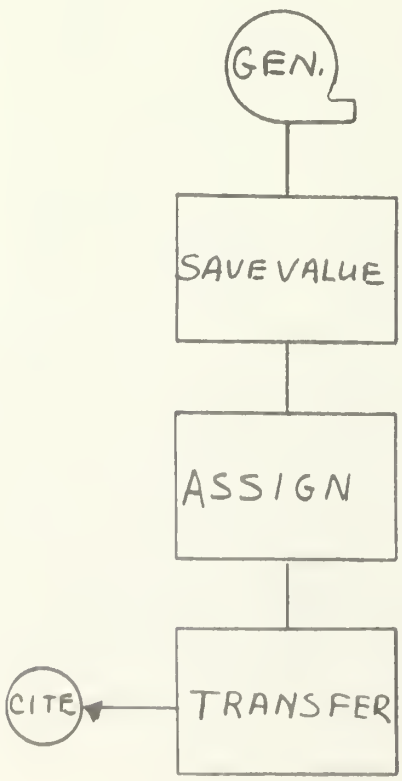


GENERATE A ROUTINE MESSAGE FOR SITE

STORE MEAN RATE OF ARRIVAL FOR NEXT MESSAGE

ASSIGN LENGTH OF MESSAGE TO PARAMETER 1

TRANSFER TO CITE

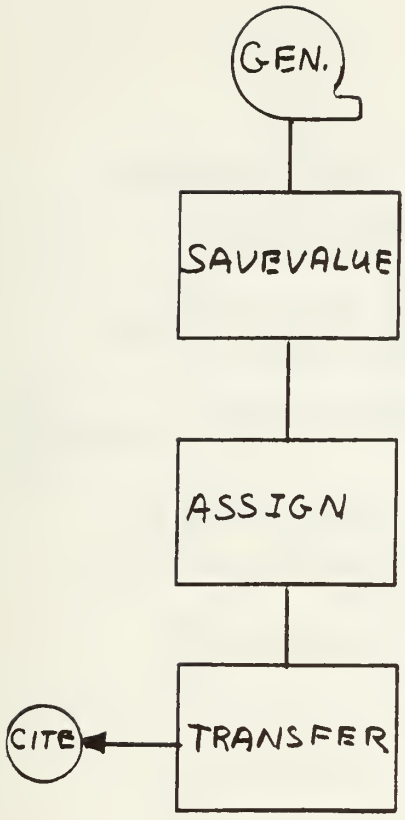


GENERATE A PRIORITY MESSAGE FOR SITE

STORE MEAN RATE OF ARRIVAL FOR NEXT MESSAGE

ASSIGN LENGTH OF MESSAGE TO PARAMETER 1

TRANSFER TO CITE

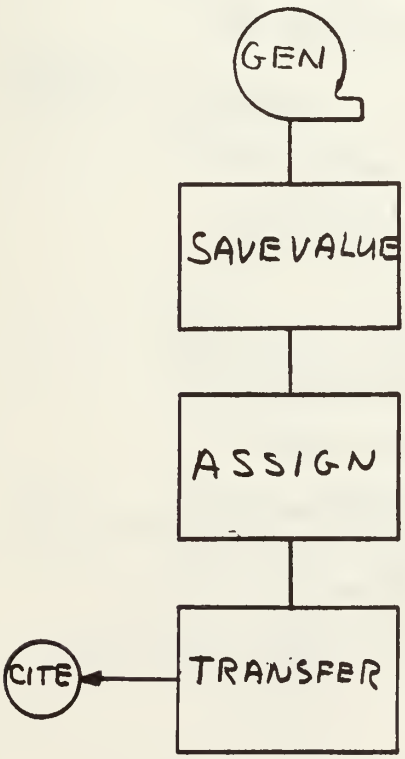


GENERATE AN IMMEDIATE MESSAGE FOR SITE

STORE MEAN RATE OF ARRIVAL FOR NEXT MESSAGE

ASSIGN LENGTH OF MESSAGE TO PARAMETER 1

TRANSFER TO CITE

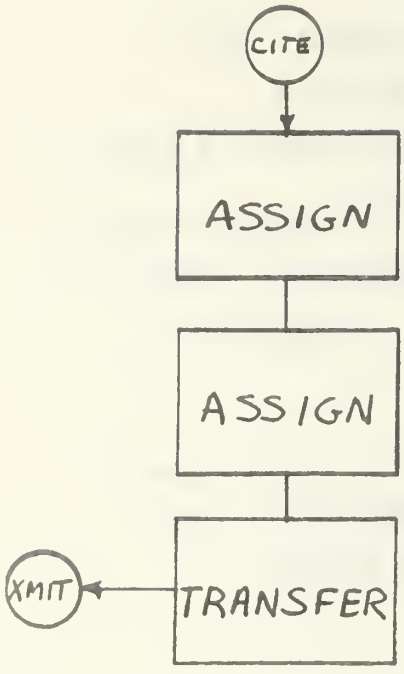


GENERATE A FLASH MESSAGE FOR SITE

STORE MEAN RATE OF ARRIVAL FOR NEXT MESSAGE

ASSIGN LENGTH OF MESSAGE TO PARAMETER 1

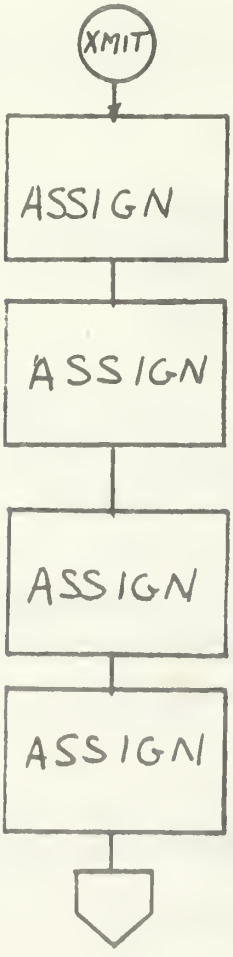
TRANSFER TO CITE



ASSIGN DESTINATION TO PARAMETER 3

ASSIGN REQUIRED ROUTING TO PARAMETER 4

TRANSFER MES'S. FROM CITES 1,2,34,5 TO XMIT

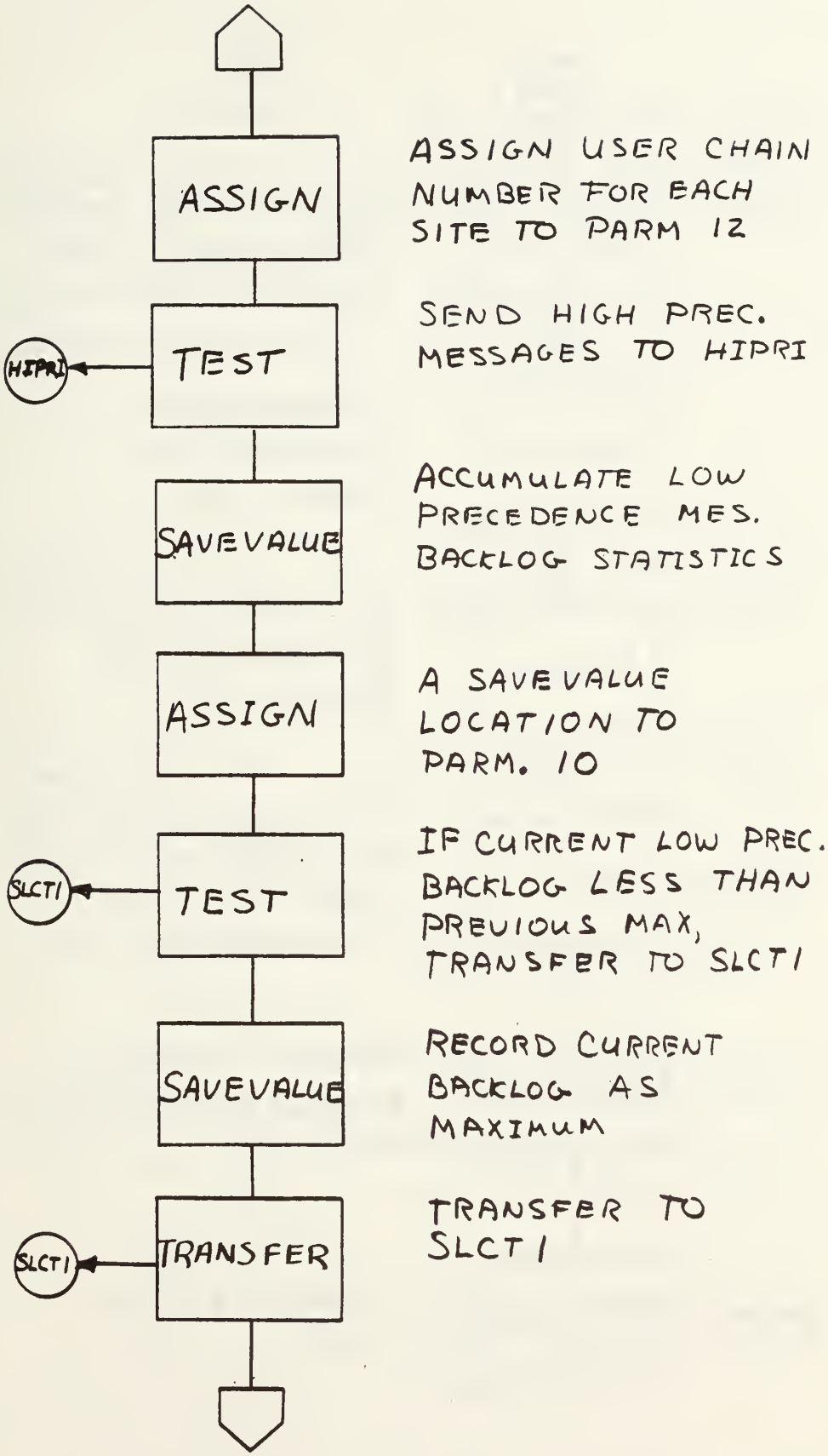


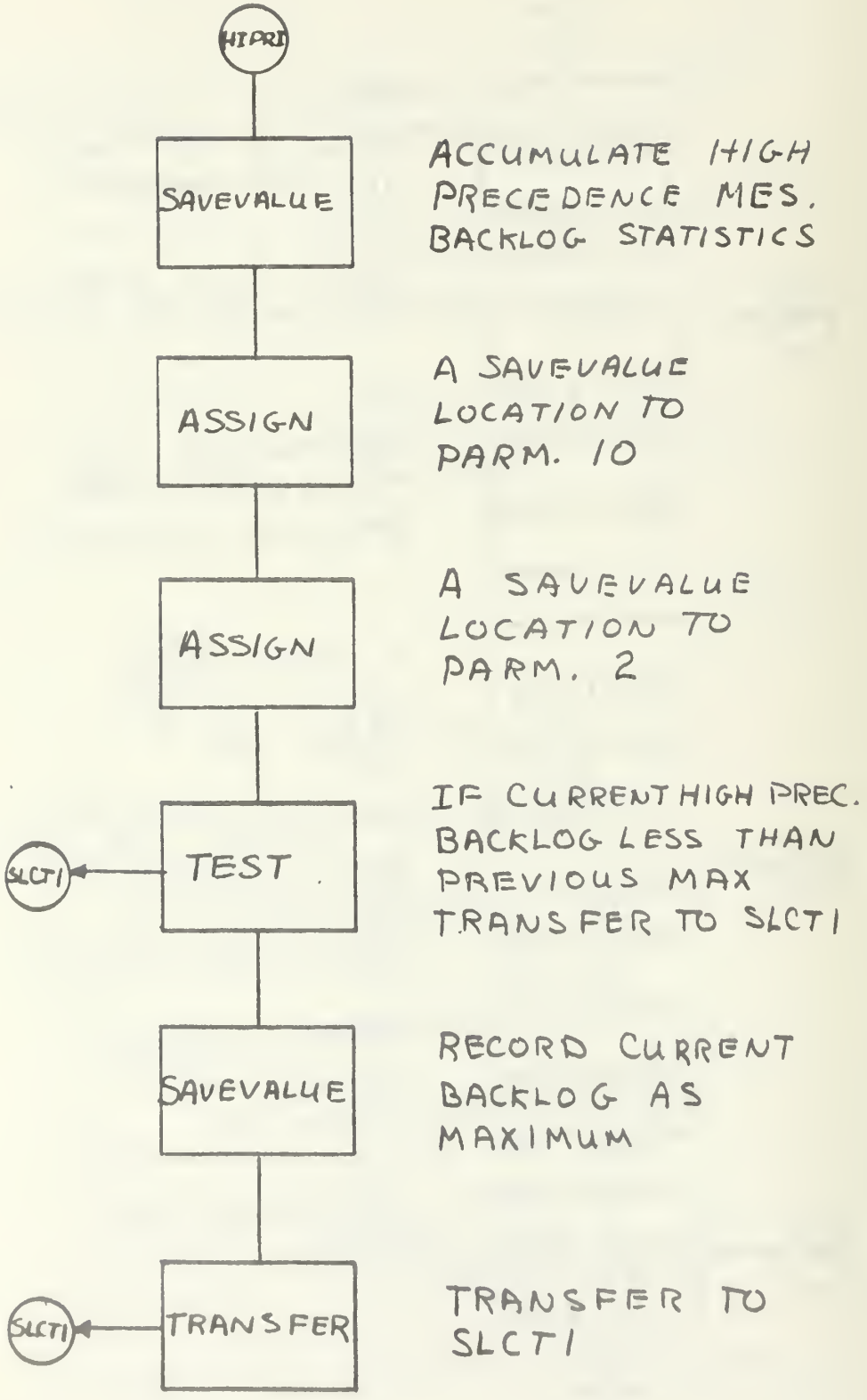
ASSIGN ROUTING FOR 1ST HOP TO PARAMETER 5

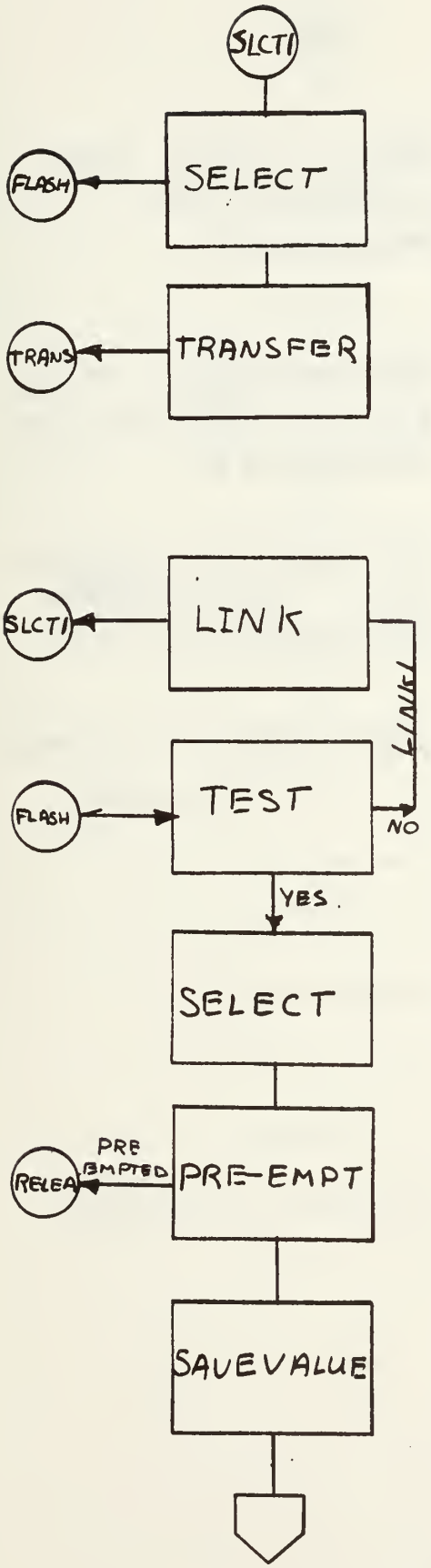
ASSIGN LOWEST XMTR. NO. FOR THIS SITE TO PARAMETER 8

ASSIGN HIGHEST XMTR NO FOR THIS SITE TO PARAMETER 9

FF PREC. = 2, ØØ = 3
PP = 4, RR = 5 and 1
RESERVD FOR PREEMPTED


ASSIGN
ASSIGN USER CHAIN NUMBER FOR EACH SITE TO PARM 12
TEST
HIPRI
SEND HIGH PREC. MESSAGES TO HIPRI
SAVEVALUE
ACCUMULATE LOW PRECEDENCE MES. BACKLOG STATISTICS
ASSIGN
A SAVEVALUE LOCATION TO PARM. 10
TEST
SLCT1
IF CURRENT LOW PREC. BACKLOG LESS THAN PREVIOUS MAX, TRANSFER TO SLCT1
SAVEVALUE
RECORD CURRENT BACKLOG AS MAXIMUM
TRANSFER
SLCT1
TRANSFER TO SLCT1


HIPRI
SAVEVALUE
ACCUMULATE HIGH PRECEDENCE MES. BACKLOG STATISTICS
ASSIGN
A SAVEVALUE LOCATION TO PARM. 10
ASSIGN
A SAVEVALUE LOCATION TO PARM. 2
SLCT1
TEST
IF CURRENT HIGH PREC. BACKLOG LESS THAN PREVIOUS MAX TRANSFER TO SLCT1
SAVEVALUE
RECORD CURRENT BACKLOG AS MAXIMUM
SLCT1
TRANSFER
TRANSFER TO SLCT1



SELECT A FREE XMTR. IF NONE AVAILABLE TRANSFER TO FLASH

TRANSFER TO AVAILABLE XMTR. GO TO TRANS

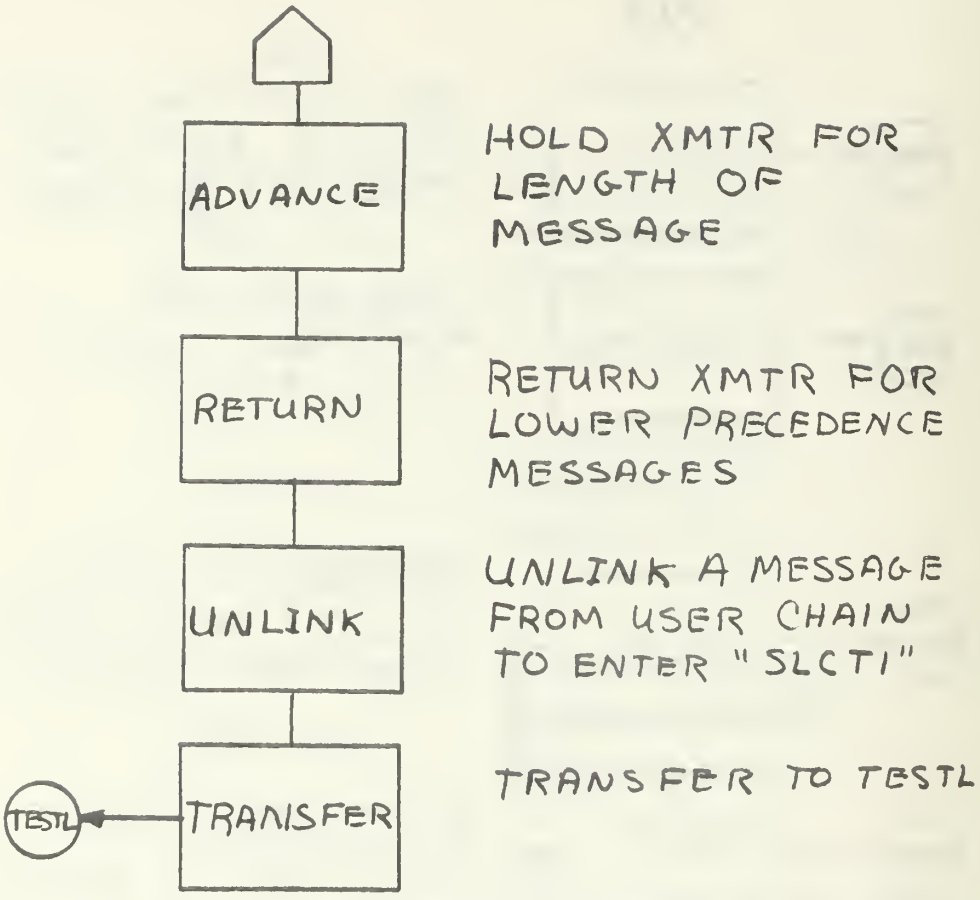
MESSAGE IS STORED IN USER CHAIN UNTIL TIME FOR XMISSION

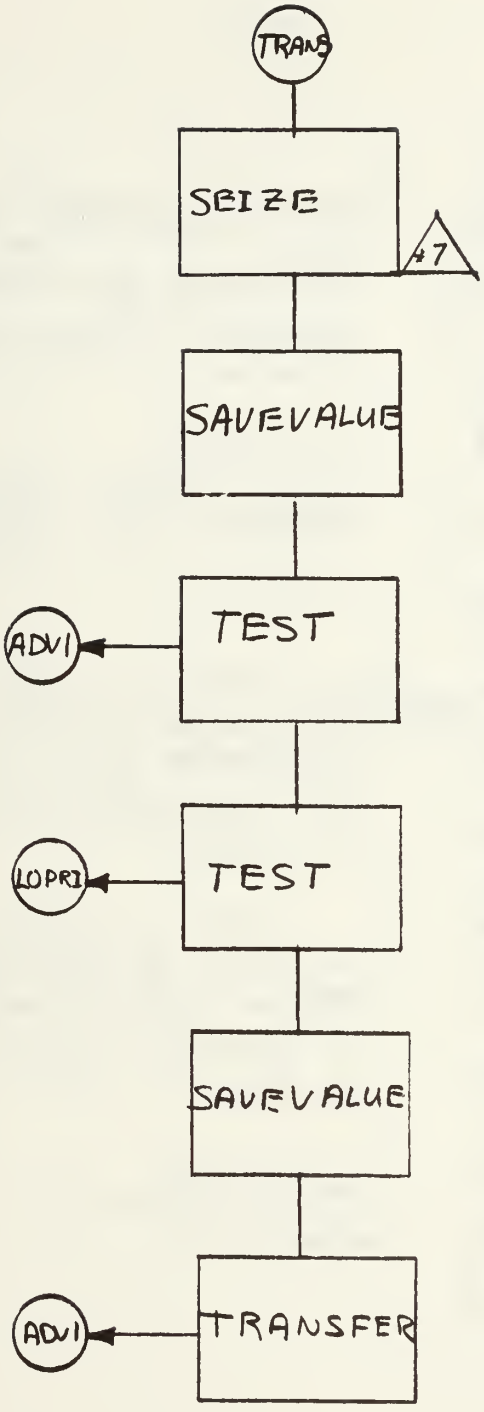
IF MESSAGE NOT FLASH, GO TO USER CHAIN (LINK1)

SELECT THE XMTR SENDING LOWEST PRECEDENCE MESS.

FLASH MESSAGE PRE-EMPT LOWEST PREC. MESSAGE. PRE-EMPTED MESSAGE GO TO USER CHAIN

DECREMENT HIGH PREC. BACKLOG BY TIME LENGTH FLASH MESSAGE


ADVANCE
HOLD XMTR FOR LENGTH OF MESSAGE
RETURN
RETURN XMTR FOR LOWER PRECEDENCE MESSAGES
UNLINK
UNLINK A MESSAGE FROM USER CHAIN TO ENTER "SLCTI"
TRANSFER
TESTL
TRANSFER TO TESTL



SEIZE A XMTR. THE NUMBER ACCORDING TO VALUE OF PARM. 7

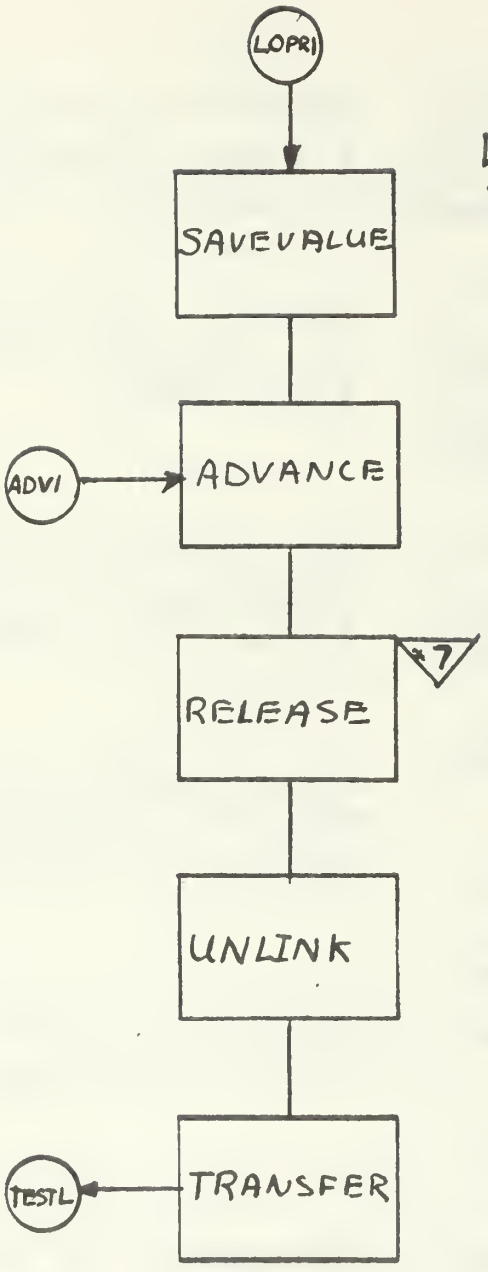
RECORD PRECEDENCE OF MESSAGE HOLDING EACH XMTR.

IF MESSAGE WAS PREVIOUSY PRE-EMPTD TRANSFER "ADV1"

IF MESSAGE IS PRIORITY OR ROUTINE TRANSFER "LOPRI"

DECREMENT HIGH PRECEDENCE BACKLOG

TRANSFER TO ADV1



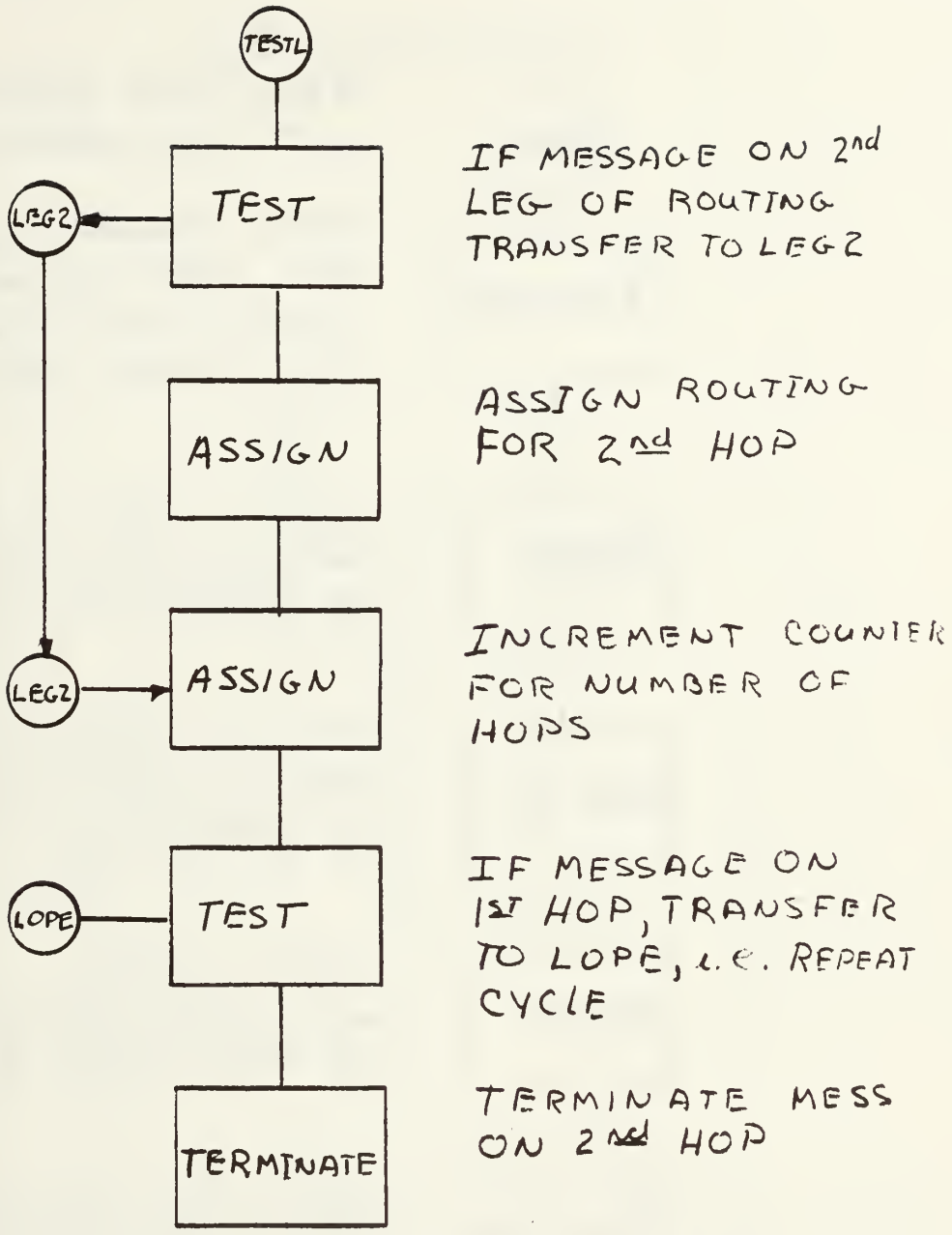
DECREMENT LO PREC BACKLOG BY TIME LENGTH FLASH MESS.

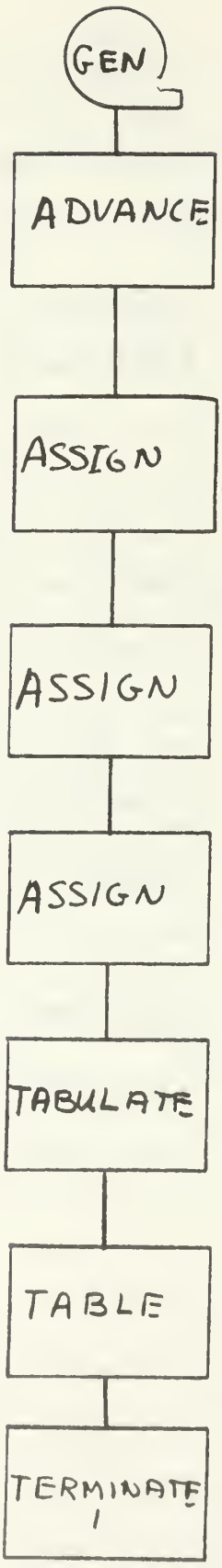
HOLD XMTR FOR LENGTH OF MESS.

RELEASE XMTR. NUMBER ACCORDING TO PARAMETER 7

UNLINK A MESSAGE FROM USER CHAIN TO ENTER BLOCK "SLCT1"

TRANSFER TO "TEST1"


TESTL
TEST
LEG2
IF MESSAGE ON 2nd LEG OF ROUTING TRANSFER TO LEG2
ASSIGN
ASSIGN ROUTING FOR 2nd HOP
LEG2
ASSIGN
INCREMENT COUNTER FOR NUMBER OF HOPS
LOPE
TEST
IF MESSAGE ON 1st HOP, TRANSFER TO LOPE, i.e. REPEAT CYCLE
TERMINATE
TERMINATE MESS ON 2nd HOP



GENERATE TRANSACTION WITH 30 PARAMETERS

HOLD TRANSACTION FOR 9000 SECONDS (LENGTH OF COMPUTER SIMULATION RUN)

ASSIGN MAX LO PREC. BACLOG STATISTICS TO PARAMETERS 1-10

ASSIGN MAX HIGH PREC. BACLOG STATS. TO PARAMETERS 11-20

ASSIGN SITE UTILIZATION STATISTICS TO PARAMETERS 21-30

TABULATE STATISTICS FROM PARAMETERS 1-30.

SPECIFICATIONS FOR OUTPUT

TERMINATE TRANSACTION AFTER EACH RUN

```
********** TCRN TAPE RELAY NETWORK PROBLEM **********
*
      REALLOCATE BLO,210,STO,0,QUE,0,LOG,0,CHA,10,GRP,0,FMS,0
      REALLOCATE HMS,0,COM,38080
*
 1    FUNCTION   V34,C6       SITE 1 PERCENT PEAK LOAD
0,.6/21600,.4/43200,.7/57600,1/72000,1/86400,.6
*
 2    FUNCTION   V34,C7       SITE 2
0,.5/14400,.5/36000,.3/57600,.7/64800,1./79200,1/86400,.5
*
 3    FUNCTION   V34,C6       SITE 3
0,.6/28800,.6/43200,1./64800,1./72000,.8/86400,.6
*
 4    FUNCTION   V34,C8       SITE 4
0,.3/14400,.3/21600,.6/43200,.6/50400,1./64800,1./79200,.5/86400,.3
*
 5    FUNCTION   V34,C7       SITE 5
0,.3/21600,.3/28800,.5/43200,.5/57600,1./72000,1./86400,.3
*
 6    FUNCTION   RN2,D4       SITE 1  DESTINATION OF MESSAGES
.2,2/.5,3/.9,4/1.,5
*
 7    FUNCTION   RN2,D4       SITE 2  DESTINATION OF MESSAGES
.25,1/.4,3/.75,4/1.,5
*
 8    FUNCTION   RN2,D4       SITE 3  DESTINATION OF MESSAGES
.4,1/.8,2/.9,4/1.,5
*
 9    FUNCTION   RN2,D4       SITE 4  DESTINATION OF MESSAGES
.2,1/.4,2/.7,3/1.,5
*
 10   FUNCTION   RN2,D4       SITE 5  DESTINATION OF MESSAGES
.2,1/.5,2/.6,3/1.,4
*
 11   FUNCTION   P3,D4        ROUTING FROM SITE 1
2,1662/3,1663/4,1664/5,1665
*
 12   FUNCTION   P3,D4        ROUTING FROM SITE 2
1,2661/3,2663/4,2664/5,2665
*
 13   FUNCTION   P3,D4        ROUTING FROM SITE 3
1,3661/2,3662/4,3664/5,3665
*
 14   FUNCTION   P3,D4        ROUTING FROM  SITE 4
1,4661/2,4662/3,4663/5,4665
*
 15   FUNCTION   P3,D4        ROUTING FROM  SITE 5
```

```
1,5661/2,5662/3,5663/4,5664
*
 16     FUNCTION   P5,D10       FACILITY NUMBERS  LOWER LIMIT
16,17/26,27/36,70/46,80/56,90/61,100/62,110/63,120/64,130/65,140
*
 17     FUNCTION   P5,D10       FACILITY NUMBERS  UPPER  LIMIT
16,22/26,33/36,73/46,84/56,94/61,105/62,114/63,124/64,136/65,143
 18     FUNCTION   P11,D4
1,5/2,4/3,3/4,2
*
 19     FUNCTION   P5,D10       USER CHAIN ARGUMENTS
16,1/26,2/36,3/46,4/56,5/61,6/62,7/63,8/64,9/65,10
*
 20     FUNCTION   P5,D10 LO PREC BACKLOG STORAGE LOCATIONS
16,390/26,391/36,392/46,393/56,394/61,395/62,396/63,397/64,398/65,399
*
 21     FUNCTION   P5,D10
16,17/26,27/36,37/46,47/56,57/61,71/62,72/63,73/64,74/65,75
*
 22     FUNCTION   P5,D10 HI PREC BACKLOG STORAGE LOCATIONS
16,380/26,381/36,382/46,383/56,384/61,385/62,386/63,387/64,388/65,389
*
 30     FUNCTION   RN1,C24      EXPONENTIAL DISTRIBUTION
0      0      .1     .104  .2     .222  .3     .355  .4     .509  .5     .69
.6     .915   .7     1.2   .75    1.38  .8     1.6   .84    1.83  .88    2.12
.9     2.3    .92    2.52  .94    2.81  .95    2.99  .96    3.2   .97    3.5
.98    3.9    .99    4.6   .995   5.3   .998   6.2   .999   7     .9997  8
*
 1      FVARIABLE  30/FN1       ARRIVAL RATE OF MESSAGES AT SITE 1
 2      FVARIABLE  70/FN1
 3      FVARIA LE  190/FN1
 4      FVARIABLE  1000/FN1
 5      FVARIABLE  40/FN2       ARRIVAL RATE OF MESSAGES AT SITE 2
 6      FVARIABLE  60/FN2
 7      FVARIABLE  90/FN2
 8      FVARIABLE  500/FN2
 9      FVARIABLE  60/FN3       ARRIVAL RATE OF MESSAGES AT SITE 3
 10     FVARIABLE  100/FN3
 11     FVARIABLE  130/FN3
 12     FVARIABLE  700/FN3
 13     FVARIABLE  50/FN4       ARRIVAL RATE OF MESSAGES AT SITE 4
 14     FVARIABLE  100/FN4
 15     FVARIABLE  140/FN4
 16     FVARIABLE  900/FN4
 17     FVARIABLE  40/FN5       ARRIVAL RATE OF MESSAGES AT SITE 5
 18     FVARIABLE  60/FN5
 19     FVARIABLE  100/FN5
 20     FVARIABLE  1000/FN5
```

```
*                                   DETERMINE AVERAGE UTILIZATION BY SITE
21      FVARIABLE   (FR17+FR18+FR19+FR20+FR21)/5
22      FVARIABLE   (FR27+FR28+FR29+FR30+FR31+FR32)/6
23      FVARIABLE   (FR70+FR71+FR72+FR73)/4
24      FVARIABLE   (FR80+FR81+FR82+FR83+FR84)/5
25      FVARIABLE   (FR90+FR91+FR92+FR93+FR94)/5
26      FVARIABLE   (FR100+FR101+FR102+FR103+FR104)/5
27      FVARIABLE   (FR110+FR111+FR112+FR113+FR114)/5
28      FVARIABLE   (FR120+FR121+FR122+FR123+FR124)/5
29      FVARIABLE   (FR130+FR131+FR132+FR133+FR134+FR135+FR136)/7
30      FVARIABLE   (FR140+FR141+FR142+FR143)/4
 31     VARIABLE    P4/100          FIND ROUTING FOR FIRST HOP
 32     VARIABLE    (P4-(V31*100))  FIND ROUTING FOR SECOND HOP
33      FVARIABLE   X*5/(60*(P9-P8+1))   HI PRECEDENCE BACKLOG
34      FVARIABLE   61000+C1            RUN COMMENCES AT 61000 SECONDS DAILY
 35     FVARIABLE   X*2/(60*(P9-P8+1))   LO PRECEDENCE BACKLOG
*
        INITIAL     X351,78/X352,175/X353,475/X354,2500
        INITIAL     X355,80/X356,120/X357,180/X358,1000
        INITIAL     X359,100/X360,166/X361,216/X362,1160
        INITIAL     X363,163/X364,333/X365,473/X366,2700
        INITIAL     X367,133/X368,200/X369,333/X370,3333
*
*
        GENERATE    X351,FN30,,,1     ROUTINE MESSAGE SITE 1
        SAVEVALUE   351,V1            ARRIVAL RATE FOR TIME OF DAY
        ASSIGN      1,100,30          MESSAGE LENGTH MEAN,EXPONENTIAL
        TRANSFER    ,CITE1
*
        GENERATE    X352,FN30,,,2     PRIORITY MESSAGE
        SAVEVALUE   352,V2
        ASSIGN      1,85,30
        TRANSFER    ,CITE1
*
        GENERATE    X353,FN30,,,3     IMMEDIATE MESSAGE
        SAVEVALUE   353,V3
        ASSIGN      1,60,30
        TRANSFER    ,CITE1
*
        GENERATE    X354,FN30,,,4     FLASH MESSAGE
        SAVEVALUE   354,V4
        ASSIGN      1,30,30
*
*
 CITE1  ASSIGN      3,FN6         DESIRED  LOCATION
        ASSIGN      4,FN11
        TRANSFER    ,XMIT         TRANSFER ALL MESSAGES TO XMIT
*
```

```
*
        GENERATE    X355,FN30,,,1     ROUTINE MESSAGE SITE 2
        SAVEVALUE   355,V5
        ASSIGN      1,140,30
        TRANSFER    ,CITE2
*
        GENERATE    X356,FN30,,,2     PRIORITY MESSAGE SITE 2
        SAVEVALUE   356,V6
        ASSIGN      1,90,30
        TRANSFER    ,CITE2
*
        GENERATE    X357,FN30,,,3     IMMEDIATE MESSAGE SITE 2
        SAVEVALUE   357,V7
        ASSIGN      1,50,30
        TRANSFER    ,CITE2
*
        GENERATE    X358,FN30,,,4     FLASH MESSAGE SITE 2
        SAVEVALUE   358,V8
        ASSIGN      1,20,30
*
*
 CITE2  ASSIGN      3,FN7          DESIRED LOCATION
        ASSIGN      4,FN12
        TRANSFER    ,XMIT          TRANSFER ALL MESSAGES TO XMIT
*
*
        GENERATE    X359,FN30,,,1     ROUTINE MESSAGE SITE 3
        SAVEVALUE   359,V9
        ASSIGN      1,120,30
        TRANSFER    ,CITE3
*
        GENERATE    X360,FN30,,,2     PRIORITY MESSAGE SITE 3
        SAVEVALUE   360,V10
        ASSIGN      1,80,30
        TRANSFER    ,CITE3
*
        GENERATE    X361,FN30,,,3     IMMEDIATE MESSAGE SITE 3
        SAVEVALUE   361,V11
        ASSIGN      1,45,30
        TRANSFER    ,CITE3
*
        GENERATE    X362,FN30,,,4     FLASH MESSAGE SITE 3
        SAVEVALUE   362,V12
        ASSIGN      1,25,30
*
 CITE3  ASSIGN      3,FN8          DESIRED LOCATION
        ASSIGN      4,FN13
        TRANSFER    ,XMIT          TRANSFER ALL MESSAGES TO XMIT
```

```
*
*
          GENERATE    X363,FN30,,,1     ROUTINE MESSAGE SITE 4
          SAVEVALUE   363,V13
          ASSIGN      1,130,30
          TRANSFER    ,CITE4
*
          GENERATE    X364,FN30,,,2     PRIORITY MESSAGE SITE 4
          SAVEVALUE   364,V14
          ASSIGN      1,100,30
          TRANSFER    ,CITE4
*
          GENERATE    X365,FN30,,,3     IMMEDIATE MESSAGE SITE 4
          SAVEVALUE   365,V15
          ASSIGN      1,60,30
          TRANSFER    ,CITE4
*
*
          GENERATE    X366,FN30,,,4     FLASH MESSAGE SITE 4
          SAVEVALUE   366,V16
          ASSIGN      1,30,30
*
*
 CITE4    ASSIGN      3,FN9             DESIRED LOCATION
          ASSIGN      4,FN14
          TRANSFER    ,XMIT             TRANSFER ALL MESSAGES TO XMIT
*
*
          GENERATE    X367,FN30,,,1     ROUTINE MESSAGE SITE 5
          SAVEVALUE   367,V17
          ASSIGN      1,100,30
          TRANSFER    ,CITE5
*
          GENERATE    X368,FN30,,,2     PRIORITY MESSAGE SITE 5
          SAVEVALUE   368,V18
          ASSIGN      1,80,30
          TRANSFER    ,CITE5
*
          GENERATE    X369,FN30,,,3     IMMEDIATE MESSAGE SITE 5
          SAVEVALUE   369,V19
          ASSIGN      1,60,30
          TRANSFER    ,CITE5
*
          GENERATE    X370,FN30,,,4     FLASH MESSAGE SITE 5
          SAVEVALUE   370,V20
          ASSIGN      1,30,30
*
 CITE5    ASSIGN      3,FN10
```

```
        ASSIGN      4,FN15
*
 XMIT   ASSIGN      5,V31          ROUTING FOR FIRST HOP
 LOPE   ASSIGN      8,FN16         LOWER LIMIT FACILITY NUMBER FOR ROUTING
        ASSIGN      9,FN17         UPPER LIMIT FACILITY NUMBER FOR ROUTING
        ASSIGN      11,PR          ASSIGN PRECEDENCE TO PARAMETER 11
        ASSIGN      11,FN18        REORDER PRECEDENCE IN INVERSE ORDER
        ASSIGN      12,FN19        USER CHAIN NUMBER BY SITE
        TEST GE     P11,4,HIPRI      ACCUMULATE STATISTICS FOR LO PREI MES.
        SAVEVALUE   *5+,P1         ACCUMULATE AMOUNT OF MESSAGE-SEC BACKLOG
        ASSIGN      10,FN22
        TEST GE     V33,X*10,SLCT1
        SAVEVALUE   *10,V33
NSTAT   TRANSFER    ,SLCT1         FIND A FREE TRANSMITTER
 HIPRI  SAVEVALUE   FN21+,P1
        ASSIGN      10,FN20
        ASSIGN      2,FN21
        TEST GE     V35,X*10,SLCT1
        SAVEVALUE   *10,V35
NSTA    TRANSFER    ,SLCT1
*
*
*
 LINK1  LINK        *12,P11,SLCT1  PUT IN USER CHAIN ACCORDING TO FN19
*
*
 SLCT1  SELECT NU   7,*8,*9,,F,FLASH SELECT A TRANSMITTER NOT IN USE
        TRANSFER    ,TRANS         TRANSFER TO AVAILABLE TRANSMITTER
*
*
 FLASH  TEST E      PR,4,LINK1   IF NOT FLASH PUT IN USER CHAIN
*
        SELECT MAX  7,*8,*9,,XH DETERMINE FACILITY PROCESSING LOWEST
*                                PRECEDENCE MESSAGE AT EACH SITE
*
        SAVEVALUE   *7,*11,H     KEEP CURRENT PRECEDENCE OF MESSAGES
        PREEMPT     *7,,RELEA,,RE     PREEMPT LOWEST PRIORITY MESSAGE
        SAVEVALUE   FN21-,P1       DECREMENT HI PRI BACKLOG
        ADVANCE     P1             ADVANCE TIME FOR MESSAGE LENGTH
        RETURN      *7             RELEASE PREEMPTING MESSAGE
        UNLINK      *12,SLCT1,1   UNLINK MESSAGE FROM USER CHAIN
        TRANSFER    ,TESTL
*
 TRANS  SEIZE       *7             SEIZE A TRANSMITTER
        SAVEVALUE   *7,*11,H       PRECEDENT OF MESSAGE FOR EACH XMTR
        TEST NE     P11,1,ADV1 PREEMPTED MES. HAS PREVIOUSLY BEEN DECRE.
        TEST LE     P11,3,LOPRI    SEGRAGATE HI PRI FROM LO PRI
        SAVEVALUE   FN21-,P1       DECREMENT HI PRI BACKLOG
```

```
          TRANSFER    ,ADV1
 LOPRI    SAVEVALUE   *5-,P1          DECREMENT LO PRI MESSAGE  BACKLOG
 ADV1     ADVANCE     P1              ADVANCE TIME FOR MESSAGE LENGTH
          RELEASE     *7            RELEASE A TRANSMITTER
          UNLINK      *12,SLCT1,1   UNLINK MESSAGE FROM USER CHAIN
          TRANSFER    ,TESTL
*
*
 RELEA    ASSIGN      11,1          ASSIGN HI PRECEDENCE-SO MESSAGE WILL BE
*                                   FIRST OUT OF USER CHAIN
          TRANSFER    ,FLASH        BACK TO USER CHAIN
*
*
*
 TESTL    TEST E      P6,KC,LEG2     TEST IF MESSAGE ON FIRST HOP
          ASSIGN      5,V32         ASSIGN ROUTING FOR SECOND HOP
 LEG2     ASSIGN      6+,1          INCREMENT NUMBER OF HOPS
          TEST GE     P6,K2,LOPE    IF ON SECOND HOP TERMINATE
          TERMINAT
*
*
*         ACCUMULATE MAX HI AND MAX LO PRECEDENCE BACKLOGS AND AVER. SITE
*         UTILIZATION EVERY 90000 SECONDS
*
 GOP      GENERATE    ,,,1,127,30
          ADVANCE     9000
          ASSIGN      1,X380
          ASSIGN      2,X381
          ASSIGN      3,X382
          ASSIGN      4,X383
          ASSIGN      5,X384
          ASSIGN      6,X385
          ASSIGN      7,X386
          ASSIGN      8,X387
          ASSIGN      9,X388
          ASSIGN      10,X389
          ASSIGN      11,X390
          ASSIGN      12,X391
          ASSIGN      13,X392
          ASSIGN      14,X393
          ASSIGN      15,X394
          ASSIGN      16,X395
          ASSIGN      17,X396
          ASSIGN      18,X397
          ASSIGN      19,X398
          ASSIGN      20,X399
          ASSIGN      21,V21
          ASSIGN      22,V22
```

```
                ASSIGN      23,V23
                ASSIGN      24,V24
                ASSIGN      25,V25
                ASSIGN      26,V26
                ASSIGN      27,V27
                ASSIGN      28,V28
                ASSIGN      29,V29
                ASSIGN      30,V30
                TABULATE    1
                TABULATE    2
                TABULATE    3
                TABULATE    4
                TABULATE    5
                TABULATE    6
                TABULATE    7
                TABULATE    8
                TABULATE    9
                TABULATE    10
                TABULATE    11
                TABULATE    12
                TABULATE    13
                TABULATE    14
                TABULATE    15
                TABULATE    16
                TABULATE    17
                TABULATE    18
                TABULATE    19
                TABULATE    20
                TABULATE    21
                TABULATE    22
                TABULATE    23
                TABULATE    24
                TABULATE    25
                TABULATE    26
                TABULATE    27
                TABULATE    28
                TABULATE    29
                TABULATE    30
1               TABLE       P1,0,1,25   MAX HI PRI BACKLOG
2               TABLE       P2,0,1,25   MAX HI PRI BACKLOG
3               TABLE       P3,0,1,25   MAX HI PRI BACKLOG
4               TABLE       P4,0,1,25   MAX HI PRI BACKLOG
5               TABLE       P5,0,1,25   MAX HI PRI BACKLOG
6               TABLE       P6,0,1,25   MAX HI PRI BACKLOG
7               TABLE       P7,0,1,25   MAX HI PRI BACKLOG
8               TABLE       P8,0,1,25   MAX HI PRI BACKLOG
9               TABLE       P9,0,1,25   MAX HI PRI BACKLOG
10              TABLE       P10,0,1,25  MAX HI PRI BACKLOG
```

```
 11     TABLE       P11,0,1,25 MAX LO PRI BACKLOG
 12     TABLE       P12,0,1,25 MAX LO PRI BACKLOG
 13     TABLE       P13,0,1,25 MAX LO PRI BACKLOG
 14     TABLE       P14,0,1,25 MAX LO PRI BACKLOG
 15     TABLE       P15,0,1,25 MAX LO PRI BACKLOG
 16     TABLE       P16,0,1,25 MAX LO PRI BACKLOG
 17     TABLE       P17,0,1,25 MAX LO PRI BACKLOG
 18     TABLE       P18,0,1,25 MAX LO PRI BACKLOG
 19     TABLE       P19,0,1,25 MAX LO PRI BACKLOG
 20     TABLE       P20,0,1,25 MAX LO PRI BACKLOG
21      TABLE       P21,600,25,16  FACILTY UTILIZATION
 22     TABLE       P22,600,25,16  FACILTY UTILIZATION
 23     TABLE       P23,600,25,16  FACILTY UTILIZATION
 24     TABLE       P24,600,25,16  FACILTY UTILIZATION
 25     TABLE       P25,600,25,16  FACILTY UTILIZATION
 26     TABLE       P26,600,25,16  FACILTY UTILIZATION
 27     TABLE       P27,600,25,16  FACILTY UTILIZATION
 28     TABLE       P28,600,25,16  FACILTY UTILIZATION
 29     TABLE       P29,600,25,16  FACILTY UTILIZATION
 30     TABLE       P30,600,25,16  FACILTY UTILIZATION
        TERMINAT    1
        START       1
        RESET       TB1-TB30
        INITIAL     X380-X399,0
        START       1,NP
        RESET       TB1-TB30
        INITIAL     X380-X399,0
        START       1,NP
        RESET       TB1-TB30
        INITIAL     X380-X399,0
        START       1,NP
        RESET       TB1-TB30
        INITIAL     X380-X399,0
        START       1,NP
        RESET       TB1-TB30
        INITIAL     X380-X399,0
        START       1,NP
        RESET       TB1-TB30
        INITIAL     X380-X399,0
        START       1
        END
```

BIBLIOGRAPHY

1. Heidorn, G.E., A Comparison of SIMSCRIPT and GPSS - III, unpublished paper, Department of Industrial Administration, Yale University, August 1966.

2. Hillier, F.S., and Lieberman, G.J., Introduction to Operations Research, Chapter 14, Holden Day, 1968.

3. International Business Machines, FORTRAN IV Language, Reference Manual, C28-6515, 8 August 1969.

4. International Business Machines, FORTRAN Programmer's Guide, Reference Manual, C28-6630, 29 March 1969.

5. International Business Machines, Messages and Codes OS/SYS/360 Operating System, Reference Manual, C28-6631, 5 December 1969.

6. International Business Machines, GPSS User's Manual, Reference Manual, H20-0326, 26 June 1968.

7. International Business Machines, GPSS Application Description, Reference Manual, H20-0186, June 1968.

8. International Business Machines, GPSS Operations Manual, Reference Manual, H20-0136, June 1968.

9. Kivat, P.J., Digital Computer Simulation: Computer Programming Languages, Memorandum RM-5883-PR, The Rand Corporation, January 1969.

10. Kivat, P.J., Digital Computer Simulation: Modeling Concepts, Memorandum RM-5378-PR, The Rand Corporation, August 1967.

11. Kivat, P.J., and Fishman, G.S., Digital Computer Simulation: Statistical Considerations, Memorandum RM-5387-PR, November 1967.

12. Naylor, T.H., and others, Computer Simulation Techniques, Wiley, April 1968.

13. Merikallio, A.E., The Past, Present and Future in General Simulation Languages, Management Science, v. 6, p. 236-267, November 1964.

14. Schriber, T.J., General Purpose Simulation System/360, Introductory Concepts and Case Studies, University of Michigan Press, Preliminary Edition, September 1968.

15. Tiechroew, D., and Lubin, F.J., Computer Simulation-Discussion of the Technique and Comparison of Languages, Communications of the ACM, v. 9, p. 723-741, October 1966.

16. Tiechroew, D., Lubin, F.J., and Truit, T.D., Discussion of Computer Simulation Techniques and Comparison of Languages, Simulation, v. 9, p. 95-98, August 1967.

17. Tocher, K.D., Review of Simulation Languages, Operational Research Quarterly, v. 16, p. 189-218, June 1965.

18. Weinert, A.E., A SIMSCRIPT-FORTRAN Cast Study, Communications of the ACM, v. 10, p. 784-793, December 1967.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department of Operations Analysis (Code 55) Naval Postgraduate School Monterey, California 93940	2
4.	Assoc. Professor Alvin F. Andrus Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
5.	Civil Schools Branch Office of Personnel Operations Washington, D. C. 20315	1
6.	Major Charles J. Petronis 159 Ashford Ave. Dobbs Ferry, New York 10522	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

REPORT TITLE

A GPSS-FORTRAN Case Study

DESCRIPTIVE NOTES (Type of report and, inclusive dates)
Master's Thesis; April 1970

AUTHOR(S) (First name, middle initial, last name)

Charles Joseph Petronis

REPORT DATE
April 1970

7a. TOTAL NO. OF PAGES
149

7b. NO. OF REFS
18

a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

0. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited

1. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Naval Postgraduate School
Monterey, California 93940

3. ABSTRACT

The objective of this study was to compare two commonly used computer simulation languages: GPSS and FORTRAN. The comparison was made by simulating an identical system in both languages. The comparison criteria used to evaluate the languages were as follows: ability to represent system, simulation time, ability to represent stocastic phenomena, programming time, computer running time, monitoring and debugging, storage requirements, data initialization and starting conditions, replication, data collection and display. The results from this study indicated that a system may be modeled four to five times faster using GPSS as compared to FORTRAN. The modeled system was simpler to conceptualize in GPSS, and the model required reduced programming, debugging and monitoring time compared to the FORTRAN model.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
GPSS						
FORTRAN						
Computer simulation						
Computer language						
Computer language comparison						

Thesis
P457
c.1
120927
Petronis
A GPSS-FORTRAN
case study.